IBM Application Program

H20-0241-2

1130 Commercial Subroutine Package

(1130-SE-25X), Version 2

Program Reference Manual

The IBM 1130 Commercial Subroutine Package is for IBM 1130 users with a knowledge of FORTRAN. The package is not intended to make FORTRAN a complete commercial language, but to supply commercial capability to users of IBM 1130 FORTRAN.

This manual is a combined user's, operator's, and system manual.

Third Edition

This edition, H20-0241-2, is a major revision of, and obsoletes, H20-0241-1.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

CONTENTS

INTRODUCTION

The IBM 1130 Commercial Subroutine Package enables the FORTRAN user to perform the basic functions of commercial programming. It provides the following commercial capabilities:

1. Floating dollar sign and asterisk check protection
2. Alphameric move and compare operations
3. Reading unformated records
4. Complete input/output character editing, with zone punch manipulation
5. Variable-length decimal arithmetic

The package is modular in design and consists of the following subroutines:

ADD - variable-length decimal add

A1DEC - conversion from A1 format to decimal format

CARRY - resolve carries in a decimal field

DECA1 - conversion from decimal format to A1 format

DIV - variable-length decimal divide

EDIT - edit a data field

FILL - fill a variable-length area with a specified character

GET - extract a data field from an input area

ICOMP - compare two variable-length decimal data fields

IOND - wait until all input/output operations are finished

KEYBD - accept characters from the keyboard

MOVE - move a variable-length alphameric data field

MPY - variable-length decimal multiply

NCOMP - compare two variable-length alphameric data fields

NSIGN - test a sign or modify a sign

NZONE - test a zone or modify a zone

PACK - conversion from A1 format to A2 format

PRINT - overlap the printing of a line on the 1132 Printer

PUNCH - punch a card on the 1442 Card Read Punch

PUT - place a variable in an output area

READ - read a card on the 1442 Card Read Punch

SKIP - skip the carriage or space lines on the 1132 Printer

STACK - select the next card to the alternate stacker on the 1442 Card Read Punch

SUB - variable-length decimal subtract

TYPER - overlap the typing of a line on the console printer

UNPAC - conversion from A2 format to A1 format

WHOLE - truncate the fraction of a real number

The 1130 Commercial Subroutine Package is designed for an IBM 1130 with 8,192 words of core storage, with card input/output, with or without the 1132 Printer, and with or without the disk storage drive.

The subroutines are written in both 1130 FORTRAN and the 1130 Assembler Language as follows:

FORTRAN	Assembler Language
ADD	IOND
A1DEC	PACK/UNPAC
CARRY	PRINT/SKIP
DECA1	READ/PUNCH
DIV	STACK
EDIT	TYPER/KEYBD
FILL	WHOLE
GET	
ICOMP	
MOVE	
MPY	
NCOMP	
NSIGN	
NZONE	
PUT	
SUB	

GENERAL DESCRIPTION

The 1130 Commercial Subroutine Package has been written to facilitate the use of FORTRAN in basic commercial programming. To accomplish this, six functions are required: (1) variable-length alphameric move, (2) variable-length alphameric compare, (3) edit, floating dollar sign, and asterisk check protection, (4) reading of unformatted records, (5) zone manipulation, and (6) variable-length decimal arithmetic.

These functions and more are supplied in the package.

The 23 subroutines making up the 1130 Commercial Subroutine Package are to be inserted in the FORTRAN execute deck or stored on the disk cartridge.

Timing for each routine is approximately 1 ms per character.

Since the routines are used in conjunction with FORTRAN and are written in FORTRAN, there are restrictions on the range of real variables. An extended precision real number must be between -1,000,000,000 and +1,000,000,000.

These restrictions do not apply to numbers in decimal format. There is no limit to the number of digits in a number that is in decimal format.

However, because the 1130 is a binary computer, a decimal fraction is not always equal to its binary equivalent. With real numbers, therefore, it is possible to have errors, called precision errors. These errors should appear in the low-order digit only. In one of the subroutines, PUT, the user has the option to half-adjust, which means that if one additional digit is carried, precision errors should not affect the results.

Therefore, the limits on a real number, dollars and cents, are:

Minimum	-1,000,000.000
Maximum	+1,000,000.000

As can be seen, an additional digit is carried for precision. In addition, all real arithmetic operations should be performed in mills, not dollars. The decimal point may be placed when results are printed.

Again, these restrictions do not apply to numbers in decimal format. There is no limit to the number of digits in a number that is in decimal format. However, all calculations with decimal numbers should also be performed with an additional digit carried (mills) This does not make it difficult to half-adjust results.

The control statement ONE WORD INTEGERS must be used in the main program in order for the subroutines to work properly. The package is being distributed in extended precision. Therefore, the control statement EXTENDED PRECISION should be used. Instructions for converting the package to standard precision are included under "Modification Aids".

In many commercial applications it is customary to X-punch the units position of a credit or negative field. Because the 11-0 Hollerith combination is not recognized by the conversion routines with FORTRAN READs, it is necessary, when keypunching, to omit the 0-punch when an 11-punch is present in the same column. This is not a problem with cards produced by the 1130, which then serve as input to subsequent runs. Any control X-punches, in any positions, will not be recognized when the underpunched digit is a zero. "Not recognized" means that the character position is replaced with a blank. This is the case for both input and output when standard FORTRAN READs and WRITEs are used.

A 12-punch is not recognized by the conversion routines with FORTRAN when the underpunched digit is a zero. Therefore, a plus zero (12-0 Hollerith) will be expressed as only a 0-punch. For this reason, plus fields should be left unzoned rather than 12-punched in the units position.

When the input routines supplied with this package are used, this problem does not exist. All zone punches are recognized and are treated properly.

With one exception, all I/O devices must use either FORTRAN I/O exclusively or Commercial Subroutine Package I/O exclusively. The exception is as follows: if the console printer uses Commercial Subroutine Package I/O, only the 1132 Printer may use either FORTRAN I/O or Commercial Subroutine Package I/O.

Because most of these subroutines are written in FORTRAN, they facilitate machine independence and modification.

MACHINE AND SYSTEM CONFIGURATION

The minimum machine configuration required to execute the 1130 Commercial Subroutine Package is as follows:

1131, Model 1B
1442 Card Read Punch, Model 6 or 7

All devices supported by FORTRAN are supported in the same manner under the 1130 Commercial Subroutine Package. In addition, the following overlap capabilities are provided:

- Printing on the 1132 Printer is overlapped with all other operations.
- Card reading on the 1442 Card Read Punch, Model 6 or 7, is overlapped with code conversion.
- Printing on the console printer is overlapped with all operations except reading from the keyboard.

With one exception, all I/O devices must use either FORTRAN I/O exclusively or Commercial Subroutine Package I/O exclusively. The exception is as follows: if the console printer uses Commercial Subroutine Package I/O, only the 1132 Printer may use either FORTRAN I/O or Commercial Subroutine Package I/O.

In order to compile the subroutines, the minimum configuration is:

1131, Model 1A
1442 Card Read Punch, Model 6 or 7

These subroutines require certain parts of the IBM 1130 Subroutine Library (see "Core Allocation" in Appendix). Provided these subroutines are available at load time, the commercial subroutine package is usable with either the Assembler Language or the FORTRAN language.

INPUT/OUTPUT CONSIDERATIONS

In general, when using the FORTRAN READ for input of data from cards, paper tape or disk, the information should be read under A1 format.

In this manner, multiple record formats can be interrogated and the data extracted.

All of the subroutines expect data in A1 format, one character per word. Therefore, cards or paper tape that are read using the FORTRAN READ statement should be read under A1 format. Since disk READs are in core image form, the data on disk can be stored in either A1 or A2 format, A2 being preferable to conserve space. There are two routines in the package, PACK and UNPAC, to convert from A1 format to A2 format, and A2 format to A1 format, respectively. These may be used in conjunction with disk input and output.

An example of reading a card under A1 format is:

```
      DIMENSION INCRD(80)
1     FORMAT(80A1)
      IO=2
      READ(IO,1) INCRD
```

Note that standard FORTRAN READ statements are not overlapped.

To write out data, which is one character per word, in A1 format, using the FORTRAN WRITE statement, A1 format should be used. If a field is purely a FORTRAN variable, the output of a line that contains this variable and information stored in A1 format may be as follows:

```
      DIMENSION INFA1(60)
1     FORMAT(F10.4,60A1)
      I=3
      WRITE(I,1) VAR,INFA1
```

where VAR is the FORTRAN variable, and INFA1 contains the A1, character, information.

Again, note that standard FORTRAN WRITE statements are not overlapped.

A part of the 1130 Commercial Subroutine Package is devoted to overlapping input/output as much as possible. For example, consider the printing of one line on the 1132 Printer:

Using the standard FORTRAN WRITE, the printing is initiated and nothing else may occur while the printing is in progress. Using the PRINT subroutine in this package,

```
CALL PRINT(IOUT,1,120,ICH12)
```

the printing is initiated and control is returned to the user's program to execute the next statement. In this way, while printing is in progress, other operations can be going on.

The following table summarizes the overlap capabilities of this package:

Device	is overlapped	with Function
Card reader		Conversion from card codes to A1 format
Console keyboard		Nothing
Card punch		Nothing
Console printer		Anything but keyboard
1132 Printer		Anything

When using any of the I/O routines in this package, the user must always place the statement CALL IOND before any STOP or PAUSE.

This will ensure that all interrupts for I/O operations have been serviced.

The use of these subroutines will speed up most commercial data processing jobs on the 1130.

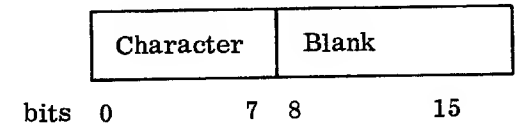
With one exception, all I/O devices must use either FORTRAN I/O exclusively or Commercial Subroutine Package I/O exclusively. The exception is as follows: if the console printer uses Commercial Subroutine Package I/O, only the 1132 Printer may use either FORTRAN I/O or Commercial Subroutine Package I/O.

SUBROUTINE PARAMETER CONSIDERATIONS

The subroutines manipulate arrays. The data contained in these arrays may be in one of three different formats: A1 format, A2 format, or decimal format.

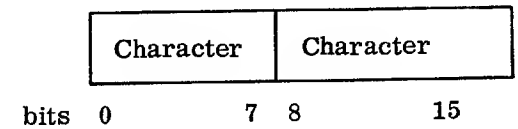
A1 format means that there is one character per 1130 word, left-justified.

A1 format:



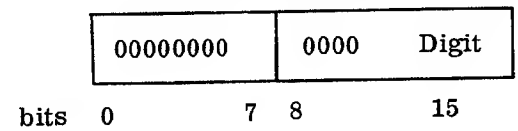
A2 format means that there are two characters per 1130 word.

A2 format:



Decimal format means that there is one decimal digit per 1130 word, right-justified.

Decimal format:



The requirements for each subroutine are as follows:

Subroutine	Format of data before processing	Format of data after processing
ADD	Decimal format	Decimal format
A1DEC	A1 format	Decimal format
CARRY	Decimal format	Decimal format
DECA1	Decimal format	A1 format
DIV	Decimal format	Decimal format
EDIT	A1 format	A1 format
FILL	Decimal constant	A1 format
GET	A1 format	Real variable
ICOMP	A1 format	Greater than, equal to, or less than zero
IOND	None	None
KEYBD	A1 format	A1 format
MOVE	A1 format	A1 format
MPY	Decimal format	Decimal format
NCOMP	A1 format	Greater than, equal to, or less than zero
NSIGN	Decimal format	Integer variable
NZONE	A1 format	Integer variable
PACK	A1 format	A2 format
PRINT	A1 format	A1 format
PUNCH	A1 format	A1 format
PUT	Real variable	A1 format
READ	A1 format	A1 format
SKIP	Decimal constant	None
STACK	None	None
SUB	Decimal format	Decimal format
TYPER	A1 format	A1 format
UNPAC	A2 format	A1 format
WHOLE	Real variable	Real variable

DETAILED DESCRIPTIONS

This section gives the general format and a description of each routine. Each description contains format, function, parameter description, detailed description, example, errors, and remarks. The function describes the capabilities of the routine. The parameter description explains in detail how the parameters, variables, and constants should be set up. The detailed description tells exactly what the subroutine does and how it should be used. Examples are given as an aid to the programmer. Certain specification and input errors may occur when using the package, and these are explained. The remarks section describes some peculiarities of the routine. Further information may be obtained from the flowcharts and listings.

ADD

Format: CALL ADD(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Sums two arbitrary-length decimal data fields, placing the result in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array which is added, the addend. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit to be added (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to be added (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the augend, the array which is added to. It will contain the result in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of KCARD (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD (the right-hand end of a field).

NER - An integer variable. Upon completion of the subroutine, this variable indicates whether arithmetic overflow occurred.

Detailed description: The corresponding digits, by place value, of JCARD and KCARD, are summed and placed back in KCARD. This operation is from left to right, with both fields being right-adjusted. Next, all carries are set in order. If overflow occurred, it is indicated by NER being equal to KLAST. NER must be initialized and reset by the user. More detailed information may be found in the ADD flowchart and listing.

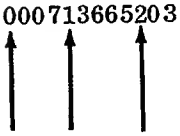
Example: DIMENSION IGRND(12),ITEM(6)

N=0

CALL ADD(ITEM,1,6,IGRND,1,12,N)

Before:

IGRND 000713665203



Position 1 5 10

ITEM 102342

Position 1 5

N=0

After:

IGRND 000713767545

Position 1 5 10

ITEM is unchanged.

N=0

The numeric data field ITEM, in decimal format, is ADDed to the numeric data field IGRND, also in decimal format. Note that the fields are both right-justified. The error indicator, N, is the same, since there is no overflow out of the high-order digit, left-hand end, of the IGRND field.

Errors: If the KCARD field is not large enough to contain the sum, that is, if there is a carry out of the high-order digit, the error indicator, NER, will be set equal to KLAST, and the KCARD field will be filled with 9s.

If the JCARD field is longer than the KCARD field, nothing will be done and the error indicator will be equal to KLAST.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine.

The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

The arithmetic performed is decimal arithmetic, using whole numbers only. No decimal point alignment is allowed. For this reason all numbers should have an assumed decimal point at the right-hand end. Dollars and cents calculations should be performed in mills so that half-adjusting, when necessary, will not be difficult. This is illustrated in the following example:

Add $1,776.00 to $2,000.07.

$1,776.00 in mills is 1776000.

$2,000.07 in mills is 2000070.

Adding, the sum is 3776070.

Half-adjusting in the mills position yields 3776075.

Using the EDIT subroutine, the result will be $3,776.07.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize and reset the error indicator.

A1DEC

Format: CALL A1DEC(JCARD,J,JLAST,NER)

Function: Converts a field from A1 format, one digit per word, to decimal format, right-justified, one digit per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the name of the field that will be converted. Originally, this field must be in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be converted (the right-hand end of a field).

NER - An integer variable. This variable will be equal to the position of the last invalid (nonnumeric or nonblank) character encountered, except for the JLAST position, which may contain a sign.

Detailed description: The subroutine operates from left to right. Each character is checked for validity (digit or blank). Blanks are changed to zeros. If a character is invalid, the error indicator, NER, is set equal to the position of the character. If the character is valid, it is converted to decimal format and right-justified using the formula

$$\text{Decimal digit} = (\text{character}+4032)/256$$

When all characters have been converted, the decimal field is signed. More detailed information may be found in the A1DEC flowchart and listing.

Example:

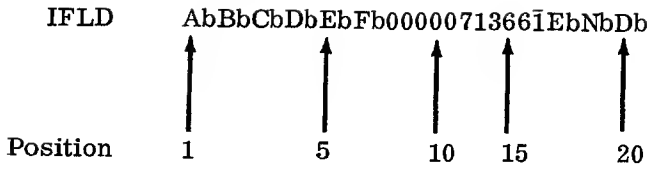
```
      DIMENSION IFLD(20)

      N=0

      CALL A1DEC(IFLD,7,17,N)
```

Before:

```
IFLD      AbBbCbDbEbFbbbbbbbbb0b7b1b3b6b6bJbEbNbDb
                ^     ^     ^     ^     ^
Position        1     5    10    15    20
```

N=0

After:



N=0

Before execution, the field is shown in A1 format, the character followed by a blank. Therefore, the field to be converted is

bbbb071366J

After execution, the field has been converted, as is evident. There were no invalid characters in the field, since N is the same.

Errors: If an invalid character (nonnumeric or nonblank) is encountered, the error indicator is set equal to the position of that character, and processing of the field continues.

Remarks: When the error indicator has been set, the character indicated is the last invalid character. There may be other invalid characters in the field, occurring to the left of the character noted.

Zone punches are used, at times, to indicate conditions (switches). These zones can be removed with the NZONE subroutine. Following is an error routine to correct errors of this type:

```
                Main Line
                    .
                    .
                    .
1       CALL A1DEC(IFLD,J,JLAST,N)
        IF(N) 2,2,3
2       Continue Main Line
                    .
                    .
                    .
3          Error Routine
        CALL NZONE(IFLD,N,4,N1)
        N1=0
        CALL A1DEC(IFLD,N,N,N1)
        IF(N1) 5,5,4
```

```
4       STOP 999

5       CALL DECA1(IFLD,J,JLAST,N)
        N=0
        GO TO 1
```

When an error of this type occurs, N will be greater than zero. Control would go to statement 3. Using the NZONE routine, the zone is removed (if not a special character). The invalid character is now converted with the A1DEC routine. If the character is still invalid, control goes to statement 4 and the program will STOP. If the character is now valid, it has been converted and control goes to statement 5. However, there may have been other invalid characters. Therefore, at statement 5 the field is converted back to A1 format and control returns to statement 1, where the field is again converted from A1 format to decimal format. This process continues until a truly invalid character (special character) is encountered, or until the field is converted with no errors.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize and reset the error indicator.

CARRY

Format: CALL CARRY(JCARD,J,JLAST,KARRY)

Function: Resolve all carries within the specified field and indicate any high-order carry out of the field. This routine will not normally be called by the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the field that will be interrogated for carries. The data must be in decimal format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of JCARD (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD (the right-hand end of a field).

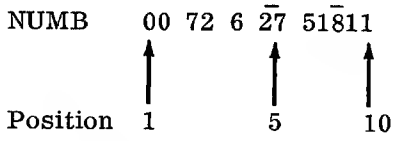
KARRY - An integer variable. This variable will contain any carry out of the high-order position of the JCARD field. If there is no carry, KARRY will be set to zero.

Detailed description: The routine operates from right to left, examining the low-order digit first. The digit being examined is divided by ten. Since only integers are used, the quotient of this division is the carry in that digit. Ten times the carry is subtracted from the digit. If the digit is now negative, ten is added to the digit and one is subtracted from the carry. At this point, or if the resultant digit was positive, the next digit to the left is examined. First, the carry from the previous digit is added to this digit. Then the process for the first digit, starting with division by ten, is carried out. When all digits have been examined, from JCARD(JLAST) to JCARD(J) inclusive, the final carry is set and the routine terminates. More detailed information may be found in the CARRY flowchart and listing.

Example:

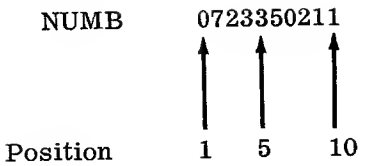
```
DIMENSION NUMB(10)
CALL CARRY(NUMB,1,10,N)
```

Before:



N=22

After:



N=0

After an arithmetic operation the condition of the NUMB field is as shown at "Before". The third, fifth and eighth positions appear as shown, because multiple arithmetic operations have generated them. The object of the CARRY routine is to resolve this type of problem.

Notice that a 1 has been borrowed from the seventh position to resolve the -8 condition. Similarly, a 3 has been borrowed from the fourth position, and the 7 from 72 has gone into the second position.

Errors: None

Remarks: This routine is used by the other routines in this package as a service routine. In general, the user need not call this routine, since all carries are resolved by the arithmetic routines themselves (ADD, SUB, MPY, DIV).

DECA1

Format: CALL DECA1(JCARD,J,JLAST,NER)

Function: Converts a field from decimal format, right-justified, one digit per word, to A1 format, one character per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the name of the field that will be converted. Originally, this field must be in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of JCARD to be converted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be converted (the right-hand end of a field).

NER - An integer variable. This variable will be equal to the position of the last digit of JCARD which was negative or greater than 9, except for the JLAST position, which can be negative (sign).

Detailed description: The subroutine operates from left to right. First the sign is determined. Then each digit, starting with JCARD(J), is converted to A1 format using the formula

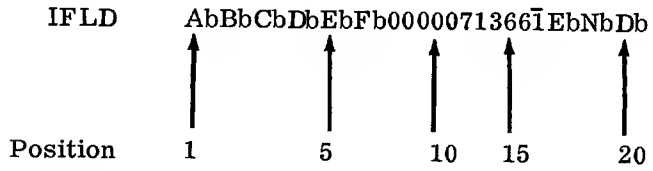
Character = 256 (decimal digit) - 4032

When all digits have been converted, the field is signed. More detailed information may be found in the DECA1 flowchart and listing.

Example:

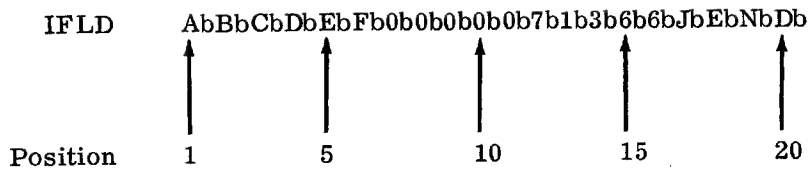
```
DIMENSION IFLD(20)

N=0

CALL DECA1(IFLD,7,17,N)
```

Before:



N=0

After:



N=0

Before execution the field is shown in decimal format. The field to be converted is

$$00000713661$$

After execution, the field has been converted to A1 format, as is evident, the character followed by a blank. There were no invalid digits in the field, since N is the same.

Errors: If an invalid digit (not 0 to 9, inclusive) is encountered, the error indicator is set equal to the position of that character, and processing of the field continues.

Remarks: When the error indicator indicates an error, the digit indicated is the last invalid digit. There may be other invalid digits in the field, occurring to the left of the digit noted.

These errors should not occur, since the arithmetic routines (ADD, SUB, MPY, and DIV) will resolve carries. However, if this does happen, the user's program should indicate (possibly by STOPing) that this has occurred.

Note that the error indicator is not reset by this subroutine. It is the responsibility of the user to initialize and reset the error indicator.

DIV

Format: CALL DIV(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Divides one arbitrary-length decimal data field by another, placing the quotient and remainder in the dividend.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array is the divisor. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the divisor (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit of the divisor (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, thedividend, will contain the quotient and the remainder, extended to the left, in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the dividend (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last digit of the dividend (the right-hand end of a field). This is also the position of the last digit of the remainder.

NER - An integer variable. Upon completion of the subroutine, this variable indicates whether division by zero was attempted, or whether the KCARD field is not long enough.

Detailed description: First the signs are cleared from both fields and saved. Then the KCARD field is extended to the left the length of the JCARD field (JLAST-J+1), and filled with zeros. If the KCARD field will be extended below KCARD(1), NER will be set equal to KLAST and the routine will be terminated. Next, the JCARD field is scanned to find the high-order significant digit. If no digit is found, the error indicator NER is set to KLAST, and the result is the same as the input. When a digit is found, the division begins. It is done by the method of trial divisors:

1. The high-order digit of the divisor is used as the trial divisor.

2. The trial divisor is divided into the next high-order digit of the dividend to generate a digit of the quotient.

3. The digit of the quotient is multiplied by the trial divisor.

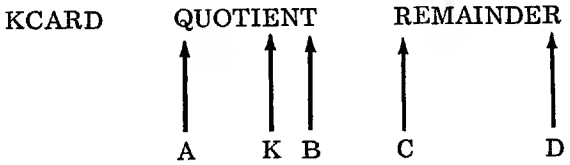
4. This product is subtracted from the corresponding number of digits in the high-order portion of the dividend.

5. As long as the result is positive, the quotient digit is the next digit in the quotient. A return is made to step 2.

6. When the result is negative, the product from step 3 is added back to the dividend, 1 is subtracted from the quotient digit, and the new quotient digit is placed in the quotient as the next digit. Finally, the signs are generated for the quotient and remainder and the sign is replaced on the divisor.

The quotient will be located in the KCARD field. The subscript of the first digit of the quotient will be K-(JLAST-J+1), and the subscript of the last digit of the quotient will be KLAST-(JLAST-J+1).

The remainder will also be located in the KCARD field. The subscript of the first digit of the remainder will be KLAST-JLAST+J, and the subscript of the last digit of the remainder will be KLAST.



A is the position whose subscript is K-(JLAST-J+1).

K is the first position of the dividend, defined earlier.

B is the position whose subscript is KLAST-(JLAST-J+1).

C is the position whose subscript is KLAST-(JLAST-J).

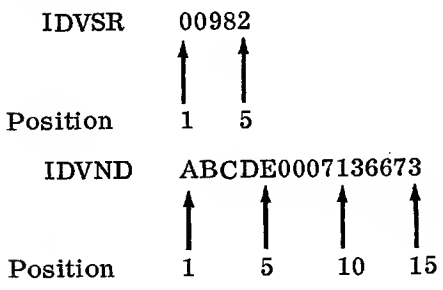
D is the position whose subscript is KLAST.

More detailed information may be found in the DIV flowchart and listing.

Example:

```
DIMENSION IDVSR(5),IDVND(15)
N=0
CALL DIV(IDVSR,1,5,IDVND,6,15,N)
```

Before:

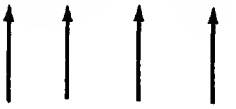


N=0

After:

IDVSR is unchanged.

IDVND 000000726700479



N=0

The numeric data field IDVND has been divided by the numeric data field IDVSR, the remainder and quotient being placed in IDVND in reverse order (quotient followed by remainder). Note that the IDVND field has been extended to the left the length of the IDVSR field, five positions.

Errors: If division by zero is attempted, the only action is that KCARD is extended and filled with zeros. The error indicator indicates that division by zero was attempted (NER=KLAST).

If there is not enough room to extend the KCARD field to the left, NER will again be set equal to KLAST, and the routine will terminate. None of the fields involved will be modified.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine.

The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

The arithmetic performed is decimal arithmetic, using whole numbers only. No decimal point alignment is allowed. For this reason numbers should have an assumed decimal point at the right-hand end. Dollars and cents calculations should be performed in mills so that half-adjusting, when necessary, will not be difficult. This is illustrated in the following example:

Divide $166.75 by 36.25 hours to find the rate per hour.

$166.75 in mills is 166750.

36.25 to three decimal places is 36250.

The units in the division are mills per one-thousandth of an hour. The answer is desired in mills per hour, so if the numerator is multiplied by 1000, the units will be mills per hour.

Dividing yields a quotient of 4600 and a zero remainder. Half-adjusting yields the rate of 4605 mills per hour or $4.60 per hour.

Space must always be provided in the KCARD field for expansion. The first position of the dividend, K, must be at least JLAST-J+1 positions from the beginning of KCARD. For example, if JCARD is seven positions, 1 through 7, the dividend in KCARD, must start at least seven positions (7-1+1=7) from the beginning of KCARD. This would have K equal to 8.

EDIT

Format: CALL EDIT(JCARD,J,JLAST,KCARD,K,KLAST)

Function: Edits data from one array into another array, which contains the edit mask.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be edited, called the source field, one character per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be edited (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be edited (the right-hand end of a field).

KCARD - The name of a one-dimensional, integer array defined in a DIMENSION statement. This is the array into which data is edited; it contains the edit mask before editing begins, stored one character per word, in A1 format, and is called the mask field.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of the edit mask (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than K. This is the position of the last character of the edit mask (the right-hand end of a field).

Detailed description: The following table gives the control characters for editing, the characters used to make up the mask, and their respective functions:

Control Character	Function
b (blank)	This character is replaced by a character from the source field.
0 (zero)	This character indicates zero suppression and is replaced by a character from the source field. The position of this character indicates the rightmost limit of zero suppression (see description of operation below). Blanks are inserted in the high-order nonsignificant positions of the field.

Control Character	Function
(decimal)	This character remains in the mask field where placed. It is considered a significant character and may not be zero-suppressed.
, (comma)	This character remains in the mask field where placed. However, if zero suppression is requested, this character will be removed if it is to the left of the last character to be zero-suppressed.
CR (credit)	These two characters can be placed in the two rightmost positions of the mask field. They are undisturbed if the source field is negative. (If the source field is positive, the characters C and R are blanked out.) In editing operations, a negative source field is indicated by an 11-zone over the rightmost character. Whether CR is blanked out or not, no data will be edited into these positions when CR is present, but rather into the edit characters to the left. The letters C and R may be used in the remainder of the edit mask, where they will be treated as normal alphabetic characters, without being subject to sign control. Only the R character is checked, so the C character may be any legal character, and it will be treated as described.
- (minus)	This character is handled similarly to CR in the rightmost position of the mask field.
* (asterisk)	This character operates the same as the 0 (zero) for zero suppression, except that asterisks rather than blanks are inserted in the high-order nonsignificant positions of the field, providing asterisk check protection.
$ (floating dollar sign)	This character has the same effect as the 0 (zero) for zero suppression, except that a $ is inserted to the left of the first significant character found, or to the left of the position that stopped the zero suppression.

The operation of the edit routine may be described in five steps:

1. Characters are placed in the mask field from the source field, moving from right to left. The characters 0 (zero), b (blank), * (asterisk) and $ (dollar sign) are replaced with characters from the source field. No other characters in the mask field are disturbed.

2. If all characters in the source field have not been placed in the mask field before the end of the mask field is encountered, the whole mask is set to asterisks and editing is terminated.

3. CR (credit) and - (minus) in the rightmost positions of the mask field are blanked if the source field is positive (does not have an 11-zone over the rightmost character).

4. The zero suppression scan starts at the left end of the mask field and proceeds left to right, replacing zeros (0), blanks (b's), and commas (,). The last position replaced will occur where the zero suppression character was located, or one position to the left of where a significant character, not zero (0), blank (b), or comma (,), occurs. If the zero suppression character was an asterisk (*), the replacement character is an asterisk. Otherwise, the replacement character is a b (blank).

5. If the zero suppression character was a dollar sign ($), a dollar sign is placed in the last replaced position in the zero suppression scan.

In order for the edit routine to work correctly and as described, five rules must be followed in creating the mask field:

1. There must be at least as many b's (blanks) in the mask field as characters in the source field.

2. If the mask field contains zero (0), asterisk (*), or dollar sign ($), zero suppression will be used and the first character in the mask field must be a b (blank).

3. The mask field must not contain more than one of the following, which may appear only once:

   0 (zero)

   * (asterisk)

   $ (dollar sign)

4. If the rightmost character in the mask field is an R, the next character to the left must be a C, in order to edit with CR (credit). Both characters will be blanked if the source field is positive. If the rightmost character in the mask field is - (minus), it will be blanked if the source field is positive.

5. All numeric, alphabetic, and special characters may be used in the mask field. All characters that do not have special meaning will be left in their original position in the mask field during the edit.

   More detailed information may be found in the EDIT flowchart and listing.

Example: There are two common methods for creating a mask field:

	Method 1	Method 2
	DIMENSION MASK(10)	DIMENSION MASK(10)
1	FORMAT(10A1)	MASK(1)=16448
	IN=2	MASK(2)=27456
	READ(IN,1)MASK	MASK(3)=16448
		MASK(4)=16448
		MASK(5)=23360
		MASK(6)=19264
		MASK(7)=16448
		MASK(8)=16448
		MASK(9)=-15552
		MASK(10)=-9920

The first method, and by far the shorter and simpler, is to read the mask field in from a data card. Note that each character requires a word of core storage. The second method is to create the mask field using the FORTRAN arithmetic statement. Still another method for creating the mask field is by using the FILL routine. These last two methods make use of the decimal equivalents of EBCDIC codes as listed in the Appendix.

The table of examples below illustrates how the EDIT routine works:

Source Field	Mask Field	Result
00123D	bb,bb$.bbCR	bbb$12.34bb
00123M	bb,bb$.bbCR	bbb$12.34CR
00123M	bb,bb$.bb-	bbb$12.34-
00123D	bb,bb$.bb-	bbb$12.34b
46426723	b,bbb,bb$.bbCR	b$464,267.23bb
00200P	b,bb*.bbCR	***20.07CR
082267139	bbb-bb-bbbb	082-26-7139
01234567	bbbb$.bbCR	**********
0AB1234	bbbbb$.bbCR	b$AB12.34bb
-12345	bb,bb$.bb-	$-,123.45b

Because the mask field is destroyed after each use, it is advisable to move the mask field to the output area and perform the edit function in the output area.

Errors: If the number of characters in the source field is greater than the number of blanks in the mask field, the mask field is filled with asterisks (*).

Remarks: If JLAST is less than or equal to J, only one character will be placed in the mask field.

In order to place a b (blank) in a specific position, the FILL routine may be uscd. In addition, the EDIT routine may be modified so that the & (ampersand) will indicate a blank in the position in which the ampersand is placed.

FILL

Format: CALL FILL(JCARD,J,JLAST,NCH)

Function: Fills an area with a specified character.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the area to be filled.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be filled (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be filled (the right-hand end of a field).

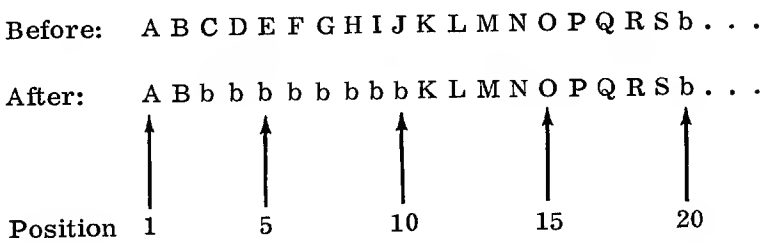
NCH - An integer constant, an integer expression, or an integer variable. This is the code for the fill character. The Appendix contains a list of these codes.

Detailed description: The area of JCARD, starting with J and ending with JLAST, is filled with the character equivalent to the NCH code, one character per word, in A1 format. More detailed information may be found in the FILL flowchart and listing.

Example: CALL FILL (IPRNT,3,10,16448)

Fill the area IPRNT from positions 3 through 10 with blanks. In other words, clear the area.

IPRNT:



Errors: None.

Remarks: If JLAST is less than J, only JCARD(J) will be filled with the character equivalent of NCH.

GET

Format: GET (JCARD,J,JLAST,SHIFT)

Function: Extracts a data field from an array, and converts it to a real number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the data to be retrieved, stored one digit per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be retrieved (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be retrieved (the right-hand end of a field).

SHIFT - A real constant, a real expression, or a real variable. If decimal places are required, SHIFT is equal to 10^{-d}, d being the number of decimal places. When SHIFT is used as a scale factor, SHIFT is 10^{d}, d being the number of zeros. If a card contains 12345 and the value of SHIFT is 0.0001, the result will be 1.2345. The result will be 123450. if a value 10.0 is assigned to SHIFT.

Detailed description: Using the formula

$$\text{BINARY DIGIT} = (\text{EBCDIC CODE} + 4032) / 256$$

the real digits are retrieved. Each binary digit is shifted left and summed, resulting in a whole number decimal. The sum is multiplied by SHIFT to locate the decimal point. The result is then placed in the real variable GET. If there are blanks in the data field, they are treated as zeros. If a nonnumeric character, other than blank, appears in any position other than the low-order position, the variable containing the result is zero. If a special character, other than the - (minus), appears in the low-order position, the resulting variable is set to zero. For input and for output the sign must be placed over the low-order position as an 11-punch for minus and a 12 or no overpunch for plus. If the low-order position is zero and the number is negative, the column must contain only an 11-punch. (The zero must not be punched.) If the low-order position is zero and the number is positive, the column must contain only the zero punch. (The 12 row must not be punched.)

More detailed information may be found in the GET flowchart and listing.

Example 1: DIMENSION INCRD(80)

B=GET(INCRD,1,5,0.001)

Before: INCRD 0123456b...

B = 0.0

After: INCRD is the same.

B = 1.234

Example 2:

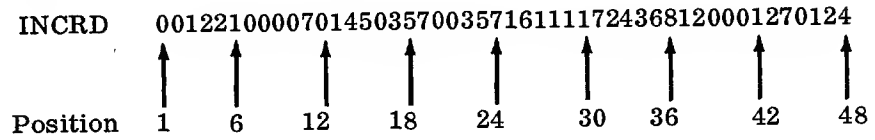
A = GET (INCRD,1,6,0.01) + GET (INCRD,7,12,0.01)

+ GET (INCRD,13,18,0.01) + GET (INCRD,19,24,0.01)

+ GET (INCRD,25,30,0.01) + GET (INCRD,31,36,0.01)

+ GET (INCRD,37,42,0.01) + GET (INCRD,43,48,0.01)

Before:



A=0.0

After: INCRD is the same

A = 21222.87

The above example sums the six-digit fields found in the first 48 columns of a card. Each data field has two decimal places. Any arithmetic operation can be performed with GET () as an operand.

Errors: If a nonnumeric character, other than blank, appears in a position other than the low-order position, the result is set to zero.

If a special character other than - (minus) appears in the low-order position, the result is set to zero.

Remarks: The GET routine is a function subprogram. As such, it is used in an arithmetic expression as shown in the example.

When the digit in the units position is a zero, a minus sign is shown as an 11-punch only; a plus is shown as a zero-punch only.

In most cases the value of SHIFT should be 10.0, placing the decimal point at the right-hand end of the number. (For dollars and cents calculations, the result of the GET would be in mills.) This will eliminate precision errors from the calculations. The decimal point may be replaced, moved to the left, with the EDIT routine for output. (See example under "Programming Notes".)

If JLAST is less than J, only one digit, JCARD(J), will be placed in the real variable GET.

ICOMP

Format: ICOMP (JCARD,J,JLAST,KCARD,K,KLAST)

Function: Two variable-length decimal format data fields are compared. The result is set to a negative number, zero, or a positive number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the first data field to be compared, one digit per word, in decimal format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be compared (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be compared (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the second data field to be compared, one digit per word, in decimal format. If the fields are unequal in length, the KCARD field must be the longer field.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to be compared (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD to be compared (the right-hand end of a field).

Detailed description: Since the fields are assumed to be right-justified, the first operation is to examine the length of each field. If KCARD is longer than JCARD, the leading digits of KCARD are examined. If any one of them is greater than zero the result (ICOMP) is the opposite sign of KCARD. If they are all zero, or if the lengths are equal, corresponding digits are compared. The routine operates from left to right. The routine terminates when KCARD is longer than JCARD and a nonzero digit appears in the high-order of KCARD, when JCARD and KCARD do not match, or when all digits in JCARD and KCARD are equal. The following table shows the value of ICOMP, depending on the relation of the JCARD field to the KCARD field:

ICOMP	Relation
- (minus)	JCARD is less than KCARD
0 (zero)	JCARD is equal to KCARD
+ (plus)	JCARD is greater than KCARD

More detailed information may be found in the ICOMP flowchart and listing.

Example:

```
DIMENSION ITOT(10),ICTL(10)

IF (ICOMP(ICTL,1,10,ITOT,1,10)) 1,2,1
```

The control total is compared to the total calculated. Control goes to statement 1 if the totals do not match (the calculated total is greater than or less than the control total). Control goes to statement 2 if the calculated total is equal to the control total. The fields compared are not changed.

ITOT	0007136673
ICTL	0007136688
ICOMP	after is positive.

Errors: No errors are detected. However, the JCARD field must not be longer than the KCARD field.

Remarks: ICOMP is a function subprogram and as such should be used in an arithmetic expression.

If JLAST is less than J, or KLAST is less than K, the result is unpredictable.

IOND

Format: CALL IOND

Function: Checks for I/O interrupts and loops until no I/O interrupts are pending.

Detailed description: The routine checks the Interrupt Service Subroutine Counter to see whether any I/O interrupts are pending. If the counter is not zero, the routine continues to check it until it becomes zero. Then the routine returns control to the user. More detailed information may be found in the IOND flowchart and listing.

Example:

```
CALL IOND

PAUSE 777
```

The two statements shown will wait until all I/O interrupts have been serviced. Then the program will PAUSE. If an I/O interrupt is pending, and IOND is not used before a PAUSE, the program will not PAUSE.

Errors: None

Remarks: This statement must always be used before a STOP or PAUSE statement.

It may also be helpful in debugging programs. Sometimes, with more than one event going on at the same time (PRINTing and processing) during debugging, difficulties can be encountered. The user may not be able to easily find the cause of trouble. The use of IOND after each I/O statement will ensure that only one I/O operation is going on at any given time.

KEYBD

Format: CALL KEYBD(JCARD,J,JLAST)

Function: Reads characters from the keyboard.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array will contain the keyed information when reading is finished. The information will be in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first word of JCARD into which a character will be keyed (the left-hand end of a field).

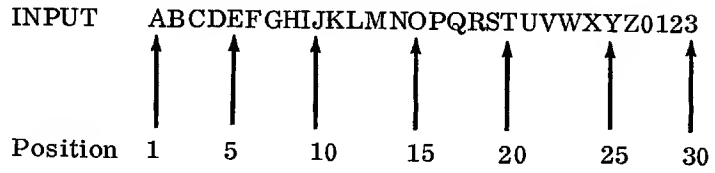
JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last word of JCARD into which a character will be keyed (the right-hand end of a field).

Detailed description: The keyboard is read and the information being read is printed on the console printer. When the specified number of characters have been read, or when EOF is encountered, the reading terminates. The characters read are converted from keyboard codes to EBCDIC and placed in A1 format, one character per word. Control is now returned to the user. More detailed information may be found in the TYPER/KEYBD flowchart and listing.

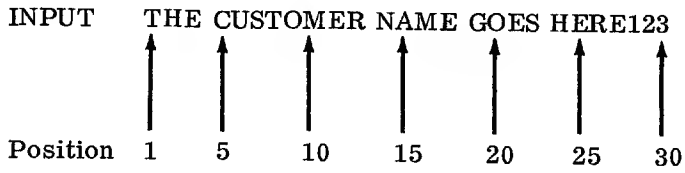
Example:

```
      DIMENSION INPUT(30)

      CALL KEYBD(INPUT,1,30)
```

Before:



After:



The array INPUT, from INPUT(1) to INPUT(30), has been filled with information read from the keyboard.

Errors: The following WAITs may occur:

WAIT (loc)	Accumulator (hex)	Action
41	2xx0	Ready the keyboard.
41	2xx1	Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.

Only 60 characters at a time may be read from the keyboard.

If JLAST is less than J, only one character will be read.

If more than 60 characters are specified (JLAST-J+1 is greater than 60), only 60 characters will be read.

Remarks: The characters asterisked in Appendix D of IBM 1130 Subroutine Library (C26-5929) will be entered into core storage and printed. All other characters will be entered into core storage but will not be printed.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O, and the 1132 Printer, which may use either FORTRAN I/O or the I/O subroutine in this package.

MOVE

Format: CALL MOVE(JCARD,J,JLAST,KCARD,K)

Function: Moves data from one array to another array.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array from which data is moved. The data must be stored in JCARD in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be moved (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be moved (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array to which data is moved, one character per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to which data will be moved (the left-hand end of a field).

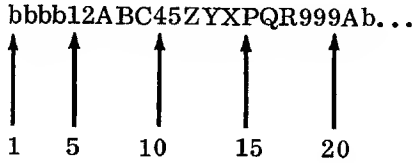
Detailed description: Characters are moved, left to right, from the sending field, JCARD, starting with JCARD(J) and ending with JCARD(JLAST), to the receiving field KCARD, starting with KCARD(K). More detailed information may be found in the MOVE flowchart and listing.

Example:

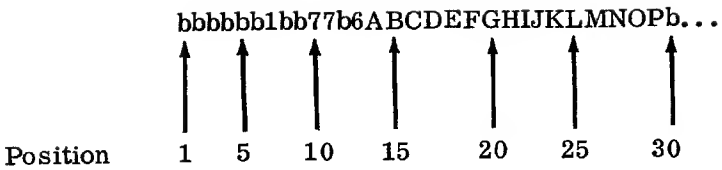
```
DIMENSION INPUT(80),IOUT(120)

L=20

K=14

CALL MOVE(INPUT,6,L,IOUT,K)
```

Before:

INPUT



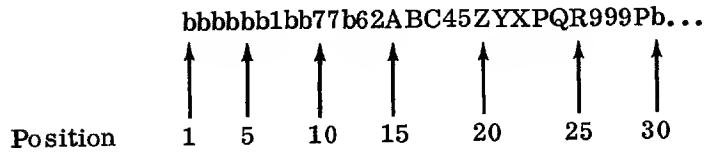
IOUT



After:

INPUT is the same.

IOUT



The field in the array INPUT, starting at INPUT(6) and ending at INPUT(20), is moved to the field in the array IOUT, starting at IOUT(14). A total of 15 characters are moved.

Errors: None

Remarks: If JLAST is less than J, one character is moved to KCARD(K).

MPY

Format: CALL MPY(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Multiplies two arbitrary-length decimal data fields, placing the product in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array is the multiplier. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit that will multiply (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to multiply (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, the multiplicand, will contain the product, extended to the left, in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of the multiplicand (the left-hand end of a field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of the product and the multiplicand (the right-hand end of a field).

NER - An integer variable. This variable will indicate whether the KCARD field is not long enough.

Detailed description: First the signs are cleared from both fields and saved. Then the KCARD field is extended to the left the length of the JCARD field (JLAST-J+1) and filled with zeros. If the KCARD field will be extended below KCARD (1), NER will be set equal to KLAST and the routine will be terminated. Next, the JCARD field is scanned to find the high-order significant digit. If no digit is found, the result is set to zero. When a digit is found, the actual multiplication begins. The significant digits in the JCARD field are multiplied by the digits in the KCARD field, one at a time, starting with KCARD(K) and ending with KCARD(KLAST). The preliminary results are summed, shifting after each preliminary multiplication to give the correct place value to the preliminary results. Finally, the correct sign is generated for the result, in KCARD, and the sign of JCARD is restored. More detailed information may be found in the MPY flowchart and listing.

Example:

```
DIMENSION MPLR(5),MCAND(15)

N=0

CALL MPY(MPLR,1,5,MCAND,6,15,N)
```

Before:

```
MPLR     00982
          ↑  ↑
Position  1  5

MCAND    ABCDE0007136673
          ↑    ↑    ↑    ↑
Position  1    5    10   15

N=0
```

After:

MPLR is unchanged.

```
MCAND    000007008212886
          ↑  ↑  ↑  ↑
Position  1  5  10 15

N=0
```

The numeric data fields MPLR and MCAND are multiplied, the result being placed in MCAND. Note that the MCAND field has been extended to the left the length of the MPLR field, five positions, and that N has not been changed.

Errors: If there is not enough room to extend the KCARD field to the left, NER will be set equal to KLAST, and the routine will terminate.

Remarks: Conversion from EBCDIC to decimal is necessary before using this subroutine. This may be accomplished with the A1DEC subroutine.

The length of the JCARD and KCARD fields is arbitrary, up to the maximum space available.

The arithmetic performed is decimal arithmetic, using whole numbers only. All numbers should have an assumed decimal point at the right-hand end. Dollars and cents

calculations should be performed in mills so that half-adjusting, when necessary, will not be difficult. This is illustrated in the following example:

Multiply 36.25 hours by $4.60.

36.25 to three decimal places is 36250.

$4.60 in mills is 4600.

Multiplying, the product is 166750000.

Half-adjusting in the mills position, adding 5 to that position only, yields 166755000.

Using the EDIT subroutine on positions 1 through 5, the result will be $166.75.

Space must always be provided in the KCARD field for expansion. The first position of the multiplicand, K, must be at least JLAST-J+1 positions from the beginning of KCARD. For example, if JCARD is 7 positions, 1 through 7, then the multiplicand, in KCARD, must start at least seven positions (7-1+1=7) from the beginning of KCARD. This would have K equal to 8.

The product, located in the KCARD field, will begin at position K-(JLAST-J+1) of KCARD, and end at position KLAST of KCARD.

NCOMP

Format: NCOMP(JCARD,J,JLAST,KCARD,K)

Function: Two variable-length data fields are compared, and the result is set to a negative number, zero, or a positive number. This is a function subprogram.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the first data field to be compared, one character per word, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be compared (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be compared (the right-hand end of a field).

KCARD - The name of a one-dimensional, integer array defined in a DIMENSION statement. This array contains the second data field to be compared, one character per word, in A1 format.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first character of KCARD to be compared (the left-hand end of a field).

Detailed description: Corresponding characters of JCARD and KCARD are compared logically, starting with JCARD(J) and KCARD(K). The routine operates from left to right. The routine terminates when JCARD and KCARD do not match, or when the character at JCARD(JLAST) has been compared. The following table shows the value of NCOMP, depending on the relation of the JCARD field to the KCARD field:

NCOMP	Relation
- (minus)	JCARD is less than KCARD
0 (zero)	JCARD is equal to KCARD
+ (plus)	JCARD is greater than KCARD

More detailed information may be found in the NCOMP flowchart and listing.

Example:

```
DIMENSION IN(80)

IF (NCOMP(IN,1,20,MASTR,1))1,2,3
```

The field on the input card starting in column 1 and ending in column 20 is compared with the master field. Control goes to statement 1 if the input card is less than the master card. Control goes to statement 2 if the input card equals the master card. Control goes to statement 3 if the input card is greater than the master card. The fields compared are not changed.

IN	1234567bbbbbbbABCDEF
MASTER	1234567bbbbbbbABCDEF
NCOMP after is zero	

Errors: None

Remarks: The collating sequence in ascending order is as follows:

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,0,1,2,3,4,5,6,7,8,9,

blank,.,<,(,+,&,*,),-,/,,,%,#,@,',=

The compare operation is terminated by the last character of the first data field, the data field at JCARD, or by an unequal comparison. NCOMP is a function subprogram and as such should be used in an arithmetic statement.

If JLAST is less than J, only the first character from each field will be compared.

NSIGN

Format: CALL NSIGN(JCARD,J,NEWS,NOLDS)

Function: Interrogate the sign and return with a code as to what the sign is. Also, modify the sign as specified.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the digit to be interrogated or modified, in decimal format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the digit to be interrogated or modified.

NEWS - An integer constant, an integer expression, or an integer variable. This is the code specifying the desired modification of the sign.

NOLDS - An integer variable. Upon completion of the routine, this variable contains the code specifying what the sign was.

Detailed description: The sign is retrieved and NOLDS is set as in the table below:

NOLDS is	When the sign was
+1	positive
-1	negative

Then a new sign is inserted, specified by NEWS, as shown in the table below:

NEWS	Sign
+1	positive
0	opposite of old sign
-1	negative
NOLDS	no change

More detailed information may be found in the NSIGN flowchart and listing.

Example: DIMENSION INUMB(9)

CALL NSIGN(INUMB,9,0,N)

Before: N=0, INUMB(9)=7

After: N=1, INUMB(9)=-7

Errors: None

Remarks: The digit processed must be in decimal format. If it is not, the results are meaningless.

NZONE

Format: CALL NZONE(JCARD,J,NEWZ,NOLDZ)

Function: Interrogate the zone and return with a code as to what the zone is. Also, modify the zone as specified.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the character to be interrogated or modified, in A1 format.

J - An integer constant, an integer expression, or an integer variable. This is the position of the character in JCARD to be interrogated or modified.

NEWZ - An integer constant, an integer expression, or an integer variable. This is the code specifying the modification of the zone.

NOLDZ - An integer variable. This variable contains the code specifying what the zone was.

Detailed description: The zone is retrieved and NOLDZ is set as in the table below:

NOLDZ is	When the character was
1	A-I
2	J-R
3	S-Z
4	0-9
more than 4	special

Then a new zone is inserted, specified by NEWZ, as shown in the table below:

NEWZ	Character
1	12 zone
2	11 zone
3	0 zone
4	no zone
more than 4	no change

When a special character is the original character, the zone will not be changed. More detailed information may be found in the NZONE flowchart and listing.

Example: DIMENSION IN(80)

CALL NZONE(IN,1,2,J)

Before: J=0,IN(1) = B

After: J=1,IN(1) = K

Errors: None

Remarks: The minus sign or dash (-, an 11-punch) is treated as if it were a negative zero, not as a special character. This is the only exception.

The only modification performed on an input minus sign is that it may be transformed to a digit zero with no zone (a positive zero).

PACK

Format: CALL PACK(JCARD,J,JLAST,KCARD,K)

Function: Information in A1 format, one character per word, is PACKed into A2 format, two characters per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the input array, containing the data in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be PACKed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than J. This is the position of the last character of JCARD to be PACKed (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is PACKed, in A2 format, two characters per word.

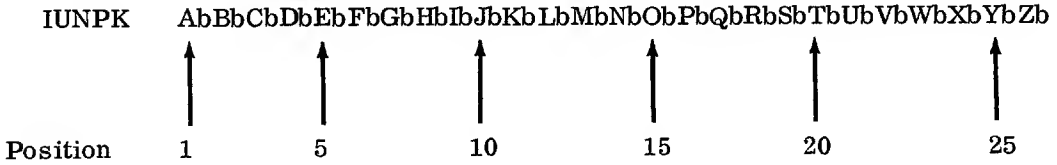
K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the PACKed characters (the left-hand end of a field).

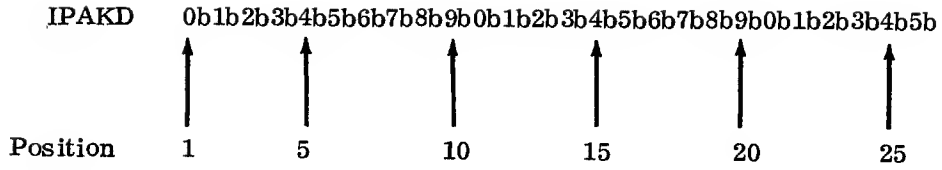
Detailed description: The characters in the JCARD array are taken in pairs, starting with JCARD(J), and PACKed together into one element of KCARD, starting with KCARD(K). Since the characters are taken in pairs, an even number of characters will always be PACKed. If necessary, the character at JCARD(JLAST+1) will be used in order to make the last data PACKed a pair. More detailed information may be found in the PACK/UNPAC flowchart and listing.

Example:

```
DIMENSION IUNPK(26),IPAKD(26)

CALL PACK(IUNPK,1,25,IPAKD,1)
```

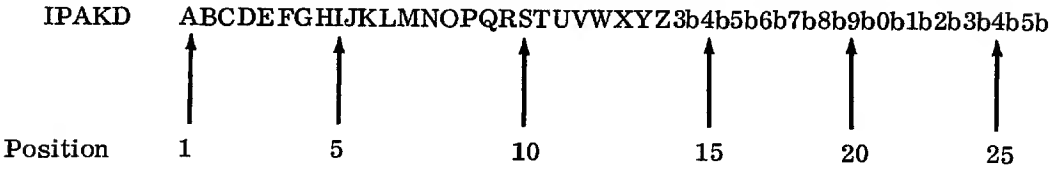
Before:





After:

IUNPK is the same.



Note that each two characters shown above represent one element of the array. Also, after IUNPK has been PACKed, the twenty-sixth character, Z, has been PACKed since 25 characters were specified (between J and JLAST).

Errors: None

Remarks: If JLAST is less than or equal to J, the first two characters of JCARD will be PACKed. An even number of characters in JCARD will always be PACKed into KCARD. An equation for how much space is required, in elements, in KCARD is

$$\text{Space in KCARD} = \left[\frac{\text{JLAST}-\text{J}+2}{2}\right]$$

This result is rounded down at all times.

PRINT

Format: CALL PRINT(JCARD,J,JLAST,NER)

Function: The printing of one line on the IBM 1132 Printer only is initiated, and control is returned to the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the information to be printed, on the IBM 1132 Printer, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be printed (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be printed (the right-hand end of a field).

NER - An integer variable. This variable indicates carriage tape channel conditions that have occurred in printing.

Detailed description: When the previous print operation is finished, if a print operation was going on, the routine begins. The characters to be printed are packed and reversed. Since the characters are taken in pairs, an even number of characters is required. If necessary, the character at JCARD(JLAST+1) will be used to get an even number. Then printing is initiated and control is returned to the user. When printing is finished, the printer spaces one line and the indicator, NER, is set as follows:

NER is	when
3	Channel 9 has been encountered
4	Channel 12 has been encountered

If channel 9 or channel 12 is not encountered, the indicator is not set.

If a WAIT occurs at location 41, one of the following conditions exists:

WAIT (loc)	Accumulator (hex)	Cause
41	6xx0	Printer not ready or end of forms.
41	6xx1	Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.

All of the above WAITs require operator intervention.

Only one line can be printed at a time (JLAST-J+1 must be less than or equal to 120).

More detailed information may be found in the PRINT/SKIP flowchart and listing.

Example:

```
      DIMENSION IOUT(120)
      N=0
      CALL PRINT(IOUT,1,120,N)
      IF(N-3) 1,2,3
2     Channel 9 routine
3     Channel 12 routine
1     Normal processing
```

The line in IOUT, from IOUT(1) through IOUT(120), is printed. The indicator is tested to see whether (1) the line was printed at channel 9 or (2) the line was printed at channel 12. Appropriate action will be taken.

Notice that the test of the indicator is made after printing. The test should always be performed in this way to see where the line has just been printed. If the indicator was set, the line was printed at channel 9 or channel 12.

Errors: If JLAST is less than J, only one character will be printed. If more than 120 characters are specified (JLAST-J+1 is greater than 120), only 120 characters will be printed.

Remarks: After each line is printed, the condition indicator should be checked for the channel 9 or channel 12 indication. In doing this the same variable should always be used for the indicator.

The indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, the FORTRAN READ and WRITE statements, except disk READ or WRITE, must not be used.

PUNCH

Format: CALL PUNCH(JCARD,J,JLAST,NER)

Function: Punches a card on the IBM 1442, Model 6 or 7, only.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the characters to be punched into a card, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be punched (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last character of JCARD to be punched (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in punching a card, and the nature of these conditions.

Detailed description: The characters to be punched are converted from EBCDIC to card codes, one at a time. When all characters have been converted, the punching operation is initiated. If an error occurs during the operation, the condition indicator is set, and the operation is continued. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or punch check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Punch not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

All of the above WAITs require operator intervention.

Only one card can be punched at a time (JLAST-J+1 must be less than or equal to 80).

More detailed information may be found in the READ/PUNCH flowchart and listing.

Example:

```
DIMENSION IOTPT(80)

N=-1

CALL PUNCH(IOTPT,1,80,N)
```

Before:

```
IOTPT     NAME...ADDRESS...AMOUNT
          ↑         ↑         ↑
Position  1         20        60
```

N=-1

After:

IOTPT is the same.

N=0

The information in IOTPT, from IOTPT(1) to IOTPT(80), has been punched into a card. Since N=0, the information was punched correctly, and the card punched into was the last card.

Errors: If a punch or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If JLAST is less than J, only one character will be punched.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be punched.

Remarks: After each card is punched, the condition indicator should be checked for the last card indication. This will occur only after the last card has physically been punched.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O, and the 1132 Printer, which may use either FORTRAN I/O or the I/O subroutine in this package.

PUT

Format: CALL PUT(JCARD,J,JLAST,VAR,ADJST,N)

Function: Converts the real variable, VAR, to an EBCDIC integer number, half-adjusting as specified, and places the result, after decimal point alignment, in an array. An 11-zone is placed over the low-order, rightmost position in the array if VAR is negative.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array will contain the result of the PUT routine, EBCDIC coded information, in A1 format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the first position of JCARD to be filled with the result (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the last position to be filled with the result (the right-hand end of a field).

VAR - A real constant, a real expression, or a real variable. This is the number to be PUT.

ADJST - A real constant, a real expression, or a real variable. This is added to the variable, VAR, as a half-adjustment factor.

N - An integer constant, an integer expression, or an integer variable. This specifies the number of digits to truncate from the right-hand end of the number, VAR.

Detailed description: First, the half-adjustment factor is added to the real variable, VAR. Then, each digit is retrieved using the formula

EBCDIC DIGIT = 256 (BINARY DIGIT) - 4032

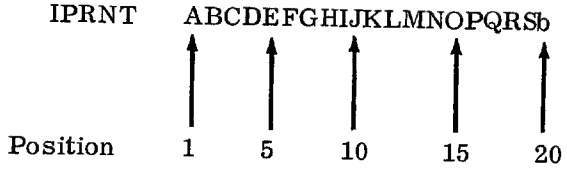
and placed in the output area. Each binary digit is retrieved by subtracting the digits already retrieved from VAR and multiplying by 10. The next digit is then retrieved and placed in the output area. More detailed information may be found in the PUT flowchart and listing.

Example:

```
DIMENSION IPRNT(120)

CALL PUT(IPRNT,1,12,A,5.0,1)
```

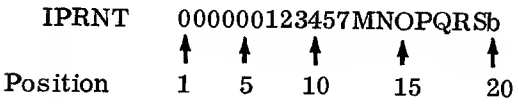
Before:

A = 1234567.



After:

A = 1234567.



Errors: None

Remarks: If the receiving field, JCARD, is not large enough to hold all of the output, only the low-order digits are placed.

It is necessary for the programmer to use the ADJST parameter in every PUT. Assume that the number to be PUT is $123.00. Because the IBM 1130 is a binary machine, the number may be represented in core storage as 122.999.... If this number is PUT with ADJST equal to zero, the result is $122.99. With ADJST equal to 0.005, the preliminary result is 123.004; when PUT, the result is $123.00. The value of ADJST should be a 5 in the decimal position one to the right of the low-order digit to be PUT.

In most cases the ADJST parameter should apply to the mills position. One digit should be specified by N (this truncates after rounding). See example under "Programming Notes".

If JLAST is less than or equal to J, only one digit will be PUT.

READ

Format: CALL READ(JCARD,J,JLAST,NER)

Function: Reads a card from the IBM 1442, Model 6 or 7, only, overlapping the conversion from card codes to EBCDIC.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. A card will be read into this array, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first word of JCARD into which a character will be read (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last word of JCARD into which a character will be read (the right-hand end of a field).

NER - An integer variable. This variable indicates any conditions that have occurred in reading a card, and the nature of these conditions.

Detailed description: A card read operation is started. While the card is being read, the characters, one at a time, are converted from card codes to EBCDIC. If an error occurs during the operation, the condition indicator is set, and the operation continues. The possible values of the condition indicator and their meaning are listed below:

NER is	when
0	Last card condition.
1	Feed or read check. Operator intervention required.

If a WAIT occurs at location 41, one of the following conditions exists:

Conditions	Accumulator (hex)
Reader not ready.	1xx0
Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your IBM representative. Save all output.	1xx1

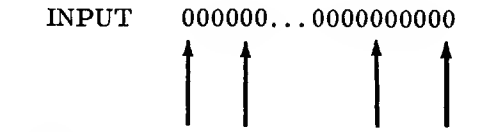
All of the above WAITs require operator intervention.

Only one card can be read at a time (JLAST-J+1 must be less than or equal to 80). More detailed information may be found in the READ/PUNCH flowchart and listing.

Example:

```
DIMENSION INPUT(160)

N1=-1

CALL READ(INPUT,1,80,N1)

N2=-1

CALL READ(INPUT,81,160,N2)
```

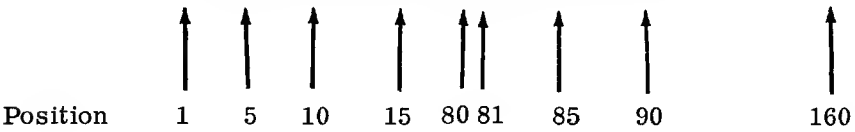
Before:



N1=-1

N2=-1

After:



N1=-1

N2=-1

From the user's viewpoint the next card is read into the INPUT array (1-80). N1 is not one of the indicated values, so the first read was successful. The next card is read into the INPUT array (81-160). N2 is not one of the indicated values, so the second read was also successful.

Errors: If a read or feed check occurs, the condition indicator will be set equal to 1. If an internal error occurs, the system will WAIT as specified above.

If JLAST is less than J, only one character will be read.

If more than 80 characters are specified (JLAST-J+1 is greater than 80), only 80 characters, one card, will be read.

Remarks: After each card read, the condition indicator should be checked for the last card indication. This will occur only after the last card has physically been read into core storage.

The condition indicator is not reset by the subroutine. It is the responsibility of the user to initialize and reset this indicator.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O, and the 1132 Printer, which may use either FORTRAN I/O or the I/O subroutine in this package.

SKIP

Format: CALL SKIP(N)

Function: Execute the requested control function on the IBM 1132 Printer only.

Parameter description:

N - An integer constant, an integer expression, or an integer variable. The value of this variable corresponds to an available control function.

Detailed description: If the printer is busy, the subroutine WAITs. Otherwise, or when the printer finishes, the routine executes the requested function and returns control to the calling program. The control functions and their values are as follows:

Function	Value
Immediate skip to channel 1	12544
Immediate skip to channel 2	12800
Immediate skip to channel 3	13056
Immediate skip to channel 4	13312
Immediate skip to channel 5	13568
Immediate skip to channel 6	13824
Immediate skip to channel 9	14592
Immediate skip to channel 12	15360
Immediate space of 1 space	15616
Immediate space of 2 spaces	15872
Immediate space of 3 spaces	16128
Suppress space after printing	0

Normal spacing is one space after printing.

Example:

```
NUMBR=12544

CALL SKIP(NUMBR)
```

The carriage skips until a punch in channel 1 of the carriage control tape is encountered (normally this is at the top of a page).

Errors: Only the codes mentioned above can be used. The use of anything else will result in either no movement of the carriage or a WAIT at location 41 with 6xx1 in the accumulator (hex).

Remarks: When space suppression after printing is executed, it is reset to single-space after printing. If the user wishes to continue suppression, he must give that skip command again.

If this subroutine is used, the FORTRAN READ and WRITE statements, except disk READ or WRITE, must not be used.

STACK

Format: CALL STACK

Function: Selects the alternate stacker on the IBM 1442, Model 6 or 7, only for the next card to go through the punch station. More detailed information may be found in the STACK flowchart and listing.

Example: A card has been read. The sum of the four-digit numbers in columns 10-13 and 20-23 is punched in columns 1-5. If the sum is negative, the card should be selected into the alternate stacker. A program to solve the problem follows:

	FORTRAN Statement	Meaning
1	FORMAT(9X,I4,6X,I4)	Description of the input data.
2	FORMAT(I5)	Description of the output data.
	IO=2	Input unit number.
3	READ(IO,1)I1,I2	Input statement.
	I3=I1+I2	Sum.
	IF(I3)4,5,5	Is the sum negative?
4	CALL STACK	Yes — select the card.
5	WRITE(IO,2)I3	No — punch.
	GO TO 3	Process the next card.
	END	

Errors: None

Remarks: If the card reader is in a not-ready state (last card) and the card just read is to be stacker-selected, the card reader will not accept the stacker select command. The user should place a blank card after the card designating last card to his program. This will prevent the card reader from becoming not ready and will allow the card to be stacker-selected.

SUB

Format: CALL SUB(JCARD,J,JLAST,KCARD,K,KLAST,NER)

Function: Subtracts one arbitrary-length decimal data field from another arbitrary-length decimal data field, placing the result in the second data field.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array that is subtracted, the subtrahend. The data must be stored in JCARD in decimal format, one digit per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first digit to be subtracted (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to J. This is the position of the last digit to be subtracted (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array, the minuend, is subtracted from, and will contain the result in decimal format, one digit per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first digit of KCARD (the left-hand end of the field).

KLAST - An integer constant, an integer expression, or an integer variable, greater than or equal to K. This is the position of the last character of KCARD (the right-hand end of a field).

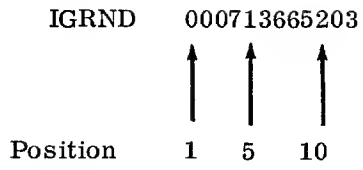
NER - An integer variable. Upon completion of the subroutine, this variable will indicate whether arithmetic overflow occurred.

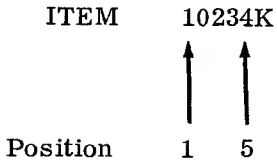
Detailed description: The sign of the JCARD field is reversed and then the JCARD and KCARD fields are ADDed using the ADD subroutine. More detailed information may be found in the SUB flowchart and listing.

Example:

```
DIMENSION IGRND(12), ITEM(6)

N=0

CALL SUB(ITEM,1,6,IGRND,1,12,N)
```

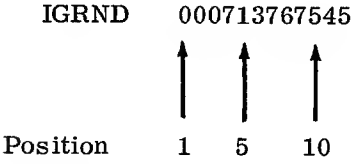
Before:





N=0

After:



ITEM is unchanged.

N=0

The numeric data field ITEM, in decimal format, is SUBtracted from the numeric data field IGRND, also in decimal format. Note that the fields are both right-justified. In this case, since the ITEM field is negative, and the operation to be performed is subtraction, the ITEM field is added to the IGRND field. The error indicator, N, is the same, since there is no overflow out of the high-order digit, left-hand end, of the IGRND field.

Errors: If the KCARD field is not large enough to contain the sum (that is, if there is a carry out of the high-order digit), the error indicator, NER, will be set equal to KLAST.

If the JCARD field is longer than the KCARD field, nothing will be done and the error indicator will be equal to KLAST.

Remarks: See the remarks for the ADD subroutine.

TYPER

Format: CALL TYPER(JCARD,J,JLAST)

Function: The typing on the console printer is initiated, and control is returned to the user.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This array contains the characters to be printed on the console printer, in A1 format, one character per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first character of JCARD to be printed (the left-hand end of a field).

JLAST - An integer constant, an integer variable, or an integer expression, greater than or equal to J. This is the position of the last character of JCARD to be printed (the right-hand end of a field).

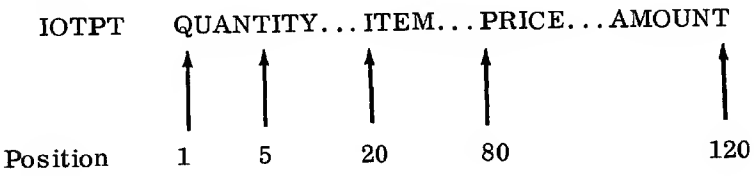
Detailed description: The characters to be printed are converted from EBCDIC to console printer codes and are packed. Since the characters are taken in pairs, an even number of characters is required. If necessary, the character at JCARD(JLAST+1) will be used to get an even number. Then the print operation is started. While printing is in progress, control is returned to the user's program.

More detailed information may be found in the TYPER/KEYBD flowchart and listing.

Example:

```
DIMENSION IOTPT(120)

CALL TYPER(IOTPT,1,120)
```

Before:



After:

IOTPT is the same. The line is being printed.

The printing of the line, specified in IOTPT, is initiated on the console printer, and control returns to the user's program.

Errors: The following WAITs may occur:

WAIT (loc)	Accumulator (hex)	Action
41	2xx0	Ready the console printer.
41	2xx1	Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listing in this manual. If the deck is the same, contact your local IBM representative. Save all output.

If JLAST is less than J, two characters will be printed. If more than 120 characters are specified (JLAST-J+1 is greater than 120), only 120 characters will be printed.

Remarks: The asterisked characters in Appendix D of IBM 1130 Subroutine Library (C26-5925) are legal. No other characters will be printed.

If this subroutine is used, any other I/O must use commercial subroutines, with the exception of disk, which must always use FORTRAN I/O, and the 1132 Printer, which may use either FORTRAN I/O or the I/O subroutine in this package.

Control functions can be used on the console printer. The following table indicates the available control functions and the decimal constant required for each function:

Function	Decimal constant
Tabulate	1344
Shift to black	5184
Carrier return	5440
Backspace	5696
Line feed	9536
Shift to red	13632

The decimal constant corresponding to a particular function must be placed in the output area (JCARD). The function will take place when its position in the output area is printed.

Example:

```
JCARD(1)=5440

JCARD(21)=1344

JCARD(30)=5440

JCARD(51)=5440

JCARD(82)=5440

CALL TYPER(JCARD,1,101)
```

The above coding will carrier-return to a new line, then print characters 2-20 of JCARD, tab to the next tab stop; print characters 22-29, carrier return, print characters 31-50, carrier return, print characters 52-81, carrier return, and finally print characters 83-101.

UNPAC

Format: CALL UNPAC(JCARD,J,JLAST,KCARD,K)

Function: Information in A2 format, two characters per word, is UNPACked into A1 format, one character per word.

Parameter description:

JCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the input array, containing the data in A2 format, two characters per word.

J - An integer constant, an integer expression, or an integer variable. This is the position of the first element of JCARD to be UNPACked (the left-hand end of a field).

JLAST - An integer constant, an integer expression, or an integer variable greater than or equal to J. This is the position of the last element of JCARD to be UNPACked (the right-hand end of a field).

KCARD - The name of a one-dimensional integer array defined in a DIMENSION statement. This is the array into which the data is UNPACked, in A1 format, one character per word.

K - An integer constant, an integer expression, or an integer variable. This is the position of the first element of KCARD to receive the UNPACked characters (the left-hand end of a field).

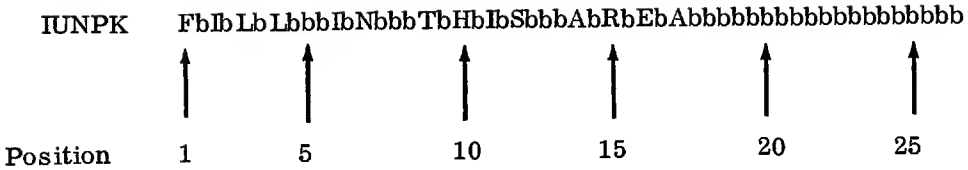
Detailed description: The characters in the JCARD array (A2) are UNPACked left to right, starting with JCARD(J), and placed in the KCARD array (A1), starting with KCARD(K). Each element of JCARD, when UNPACked, will require two elements of KCARD. More detailed information may be found in the PACK/UNPAC flowchart and listing.

Example:

```
DIMENSION IUNPK(26),IPAKD(26)

CALL UNPAC(IPAKD,1,13,IUNPK,1)
```

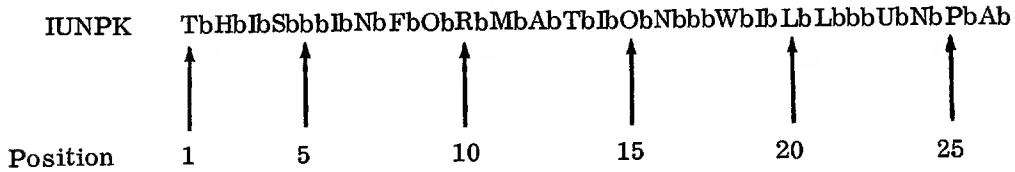
Before:

```
IPAKD     THISbINFORMATIONbWILLbUNPACKEDbbbbbbbbbbbbbbbbbbbbb
                ↑     ↑     ↑     ↑     ↑     ↑
Position        1     5     10    15    20    25
```



After:

IPAKD is the same.



Note that each two characters shown above represent one element of the array.

Errors: None

Remarks: If JLAST is less than or equal to J, only the first element of JCARD,JCARD(J) will be UNPACked into the first two elements of KCARD. An even number of characters will always be UNPACked into KCARD. An equation for how much space is required, in elements, in KCARD is

$$\text{Space in KCARD} = 2\ (\text{JLAST}-\text{J}+1)$$

WHOLE

Format: WHOLE (EXPRS)

Function: Truncates the fractional portion of a real expression.

Parameter description:

EXPRS - A real expression. This is the expression that is truncated (the fractional part is made zero).

Detailed description: The result of the expression is shifted right until the fractional portion has been shifted off. Then the result is shifted left to give the original result with a zero fraction.

Example: A=WHOLE(.1*B)

Before:

A=0.0

B=71234.99

After:

A=7123.000

B=71234.99

The expression, (.1*B), has been evaluated, and the fractional portion has been dropped.

Errors: None

Remarks: The argument, EXPRS, must always be a real expression. If the purpose is to simply truncate the fraction from a number A, the expression must be (1.0*A).

PROGRAMMING NOTES

The 1130 Commercial Subroutine Package expects all alphameric information to be one character per word. Thus, input of data, when using standard FORTRAN READ statements, should be similar to the following:

```
      DIMENSION I(80)
1     FORMAT(80A1)
      IN=2
      READ(IN,1)I
```

The above coding will read 80 characters of information, an 80-column card, placing each character in an 1130 word. The input data is now available to the user for any and all processing. Standard FORTRAN WRITE statements for information that was READ under A1 format should also be under A1 format.

Input of data, when using the READ subroutine, will automatically be in A1 format. Also, PRINTing, PUNCHing, and TYPEing assume A1 format.

Before any STOP or PAUSE, the user must always place the statement

```
CALL  IOND
```

This will ensure that all interrupts for input/output operations have been serviced.

In order to test for a channel 9 or a channel 12 indication from the 1132 Printer, the user should place the test after the CALL to the PRINT subroutine.

```
      N=0
      CALL PRINT(IOUT,1,120,N)
      IF(N-3) 1,2,3
1     No indication
2     Channel 9 is now being printed on
3     Channel 12 is now being printed on
```

If the test is not placed directly after the CALL to PRINT, erroneous conditions may be indicated.

With one exception, all I/O devices must use either FORTRAN I/O exclusively or Commercial Subroutine Package I/O exclusively. The exception is as follows: if the console printer uses Commercial Subroutine Package I/O, only the 1132 Printer may use either FORTRAN I/O or Commercial Subroutine Package I/O.

Also, the IOCS control record should not reference any devices other than disk, since this will cause subroutines, which will not be used, to reside in core storage. All parameters required by each subroutine must be supplied when programming, or results will be erroneous.

If the user wishes to use the TRACE facilities of FORTRAN, he must use standard FORTRAN READs and WRITEs. After the TRACE facilities have served their purpose, the FORTRAN READs and WRITEs should be converted to CALLs to the I/O subroutines supplied with this package.

All programs using the 1130 Commercial Subroutine Package must be compiled with the control statement * ONE WORD INTEGERS.

The package has been prepared with Extended Precision. There are notes under "Modification Aids" for the user who wishes to use standard precision. When using the Extended Precision package, the user's program must also be compiled with the control statement * EXTENDED PRECISION.

One very useful technique involves the NZONE subroutine. It is possible to have a five-way switch by coding as follows:

```
      CALL NZONE(ISWT,1,5,I)
      IF(I-5) 2,1,1
1     The switch is a special character
2     GO TO (3,4,5,6),I
3     The switch has a 12 zone
4     The switch has an 11 zone
5     The switch has a 0 zone
6     The switch has no zone
```

If each of the possible zones is expanded by actually using the digit of the switch, it becomes a 38-way switch.

In order to move a zone from one character to another, the following coding can be used:

```
      CALL NZONE(ICH1,1,5,J)
      CALL NZONE(ICH2,1,J,I)
```

The character at ICH2(1), unless it was a special character, now has the zone of the character at ICH1(1).

Also, NSIGN may be used to move signs from one field to another.

```
CALL NSIGN(IFLD1,LAST,N,N)
CALL NSIGN(IFLD2,IEND,N,I)
```

When using the disk cartridge for storage of data, it is suggested that all data to be used in FORTRAN arithmetic statements be converted to real format. All alphameric information should be PACKed before it is written onto the disk. Decimal information should be converted to A1 format and then PACKed before it is written onto the disk. These methods will allow more information to be stored on the disk cartridge.

Half-adjusting, as explained in the description of the PUT routine, is very important to the accuracy of calculations. To be completely safe (that is, to write programs so that precision does not become a problem), the program should perform all arithmetic operations in mills. Then use the PUT routine to half-adjust and truncate the mills position. The EDIT routine may then be used to place the decimal point and any other edit character. An example is as follows:

	FORTRAN Statements	Meaning
	DIMENSION IN(80),IOUT(9),ITMP(7)	Allocate storage
1	FORMAT(80A1)	Describe input
2	FORMAT(9HbGROSS IS,9A1)	Describe output
	IREAD=2	Input unit
	IWRIT=3	Output unit
	READ(IREAD,1)IOUT	Read edit mask (bbbbb$.bb)
	READ(IREAD,1)IN	Input
	RATE=GET(IN,30,34,1.0)	Extract rate (cc 30-34) already in mills
	HRS=GET(IN,40,43,10.0)	Extract hours (cc 40-43) and add a zero to make them mills
	CURR=RATE*HRS	Calculate current earnings (now in thousandths of mills)
	CURR=WHOLE((CURR+500.0)/1000.0)	Half-adjust, make current earnings mills, and truncate any fraction
	GROSS=CURR+GET(IN,20,26,10.0)	Extract old gross (cc 20-26) making mills, and calculate new gross
	CALL PUT(ITMP,1,7,GROSS,5.0,1)	Half-adjust, truncate, and convert to A1 format
	CALL EDIT(ITMP,1,7,IOUT,1,9)	Place decimal point and dollar sign
	WRITE(IWRIT,2)IOUT	Print
	CALL EXIT	End of job
	END	

The above program will calculate gross pay. If there is an error in keypunching a field, the GET statement for that field will be zero. The GET routine can be changed and the computer made to stop, if a PAUSE, as stated under "Modification Aids", is appropriately placed.

As mentioned under "General Description", precision errors can be a problem. Therefore, the following limits are set on the size of real numbers:

+100,000,000.0

-100,000,000.0

If dollars and cents are used, the limits are:

+1,000,000.000

-1,000,000.000

As can be seen, an additional decimal place is carried to ensure accuracy.

In mills (including the additional decimal place) the limits are:

+1,000,000,000.

-1,000,000,000.

When using the decimal arithmetic feature of the 1130 Commercial Subroutine Package, it is not necessary to half-adjust to compensate for the binary nature of the 1130. However, when multiplication or division is involved, it is necessary to half-adjust to get to the nearest penny. This may be done by adding a constant of 5 to the mills position.

To truncate a field (zero out part of the fraction), the user can employ the FILL subroutine. The following statements show the multiplication of the hours worked (to two decimal places) by the rate (to three decimal places), half-adjusting in the third decimal place:

	Statement	Meaning
	DIMENSION IRATE(5),IHRS(4),IWORK(9), IGROS(6),IFIVE(1)	Allocate storage
	N=0	Initialize
	IFIVE(1)=5	
	CALL MOVE(IHRS,1,4,IWORK,6)	Set up work area
	CALL MPY(IRATE,1,5,IWORK,6,9,N)	Multiply
	IF(N) 2,1,2	Overflow?
1	CALL ADD(IFIVE,1,1,IWORK,1,7,N)	Half-adjust in mills
	CALL MOVE(IWORK,1,6,IGROS,1)	Place result and truncate
	.	
	.	
	.	
C	OVERFLOW CONDITION	
2	STOP 777	

Remember that MPY and DIV both require the extension of the second field in the operation. Also, the result may be located in the second field through the formulas given in the specific subroutine descriptions.

There are no limits to the size of numbers when the decimal feature of the package is used.

MODIFICATION AIDS

Since the source language of the subroutine package is mainly FORTRAN, modification is a relatively easy problem, provided the modification is well defined.

In the listings there are comments as to where pauses could be conveniently placed to stop on error conditions.

The following FORTRAN program may be used on an IBM 1130 or other machine to produce the decimal equivalents of character codes. The only changes to the program may be the input and output unit numbers in statements 3 and 4, and the integer width in statement 2. The program reads a card which should contain up to 80 legal characters for that machine, and prints the character and its decimal equivalent.

```
      DIMENSION N(80)
1     FORMAT(80A1)
2     FORMAT(1X,A1,1X,I6)
3     READ(2,1)N
4     WRITE(3,2)(N(I),N(I), I=1,80)
      STOP
      END
```

The package has been prepared with Extended Precision. If the user wishes to use standard precision, he must, before compiling the routines, remove cards numbered:

CSP00060	CSP03310
CSP00280	CSP03750
CSP00440	CSP04340
CSP01080	CSP04690
CSP01850	CSP04920
CSP02150	CSP05190
CSP02480	CSP05460
CSP02760	CSP09830
CSP03130	CSP12860

CSP14940

In addition, the user must change the PUT subroutine by replacing card number CSP01990 with the following six cards:

```
                                                          (cc 73-80)
      JTEST=IFIX(DIGS-10. 0*DIGT)                         CSP01982
11    IF(JTEST-10)9,10,10                                 CSP01985
10    JTEST=JTEST-10                                      CSP01988
      DIGT=DIGT+1. 0                                      CSP01991
      GO TO 11                                            CSP01994
 9    JCARD(JNOW)=256*JTEST-4032                          CSP01997
```

SAMPLE PROBLEMS

PROBLEM 1

This program has been written to exercise each of the routines. A card is read and a code on that card initiates the operation of the specified routine. The card image is printed, before execution of the routine; the resulting variable is printed, if such a variable is associated with the routine; and the card image is printed, after execution of the routine.

If the user's system has an 1132 Printer, switch 0 on the console must be in the up position and all other switches in the down position. If the user's system does not have an 1132 Printer, all switches on the console must be in the down position.

Sample Problem 1: Source Program

```
// FOR                                                                    CSP09780


                                                                          PAGE 01

** SAMPLE PROBLEM 1                                                      CSP09790
* NAME SMPL1                                                             CSP09800
* IOCS(CARD,TYPEWRITER,1132 PRINTER)                                     CSP09810
* ONE WORD INTEGERS                                                      CSP09820
* EXTENDED PRECISION                                                     CSP09830
* LIST ALL                                                               CSP09840


                                                                          PAGE 02
 SAMPLE PROBLEM 1
C-----GENERAL PURPOSE 1130 COMMERCIAL SUBROUTINE PACKAGE TEST PROGRAM.    CSP09850
      DIMENSION NCARD(80), NAMES(5,13)                                    CSP09860
1     FORMAT (80A1)                                                       CSP09870
2     FORMAT (I10, 4F10.0, F10.3)                                         CSP09880
3     FORMAT (30H0NOW TESTING 1130 CSP ROUTINE ,5A1,16H WITH PARAMETERS,  CSP09890
     X4F10.5, F10.3)                                                      CSP09900
4     FORMAT (13H CARD BEFORE=,80A1)                                      CSP09910
5     FORMAT (13H CARD AFTER =,80A1)                                      CSP09920
6     FORMAT(1H ,5I3,2X,12HCARD AFTER =,1X,80A1)                          CSP09930
7     FORMAT(1H0,4X,10HINDICATORS,3X,12HCARD BEFORE=,1X,80A1)             CSP09940
8     FORMAT (10H ANSWER IS, F20.3)                                       CSP09950
C-----DEFINE UNIT NUMBERS OF I/O DEVICES.                                 CSP09960
      CALL DATSW(0,N)                                                     CSP09970
      NREAD=2                                                             CSP09980
      NWRIT=2*(1/N)+1                                                     CSP09990
      READ (NREAD,1) NAMES                                                CSP10000
10    READ (NREAD,2) N, V1, V2, V3, V4, VAR                               CSP10010
      IF (N) 98,98,99                                                     CSP10020
98    STOP                                                                CSP10030
99    WRITE (NWRIT,3) (NAMES(I,N), I=1,5), V1, V2, V3, V4, VAR            CSP10040
      N1=V1                                                               CSP10050
      N2=V2                                                               CSP10060
      N3=V3                                                               CSP10070
      N4=V4                                                               CSP10080
      NVAR=VAR                                                            CSP10090
      NER1=0                                                              CSP10100
      NER2=0                                                              CSP10110
      NER3=0                                                              CSP10120
      NER4=0                                                              CSP10130
      NER5=0                                                              CSP10140
      READ (NREAD,1) NCARD                                                CSP10150
      IF(N-7)  21,21,22                                                   CSP10160
21    WRITE(NWRIT,4) NCARD                                                CSP10170
C-----GO TO 1130 CSP ROUTINE                                              CSP10180
      GO TO (11,12,13,14,15,16,17), N                                     CSP10190
C-----COMP ROUTINE                                                        CSP10200
11    ANS=NCOMP(NCARD,N1,N2,NCARD,N3)                                     CSP10210
      GO TO 19                                                            CSP10220
C-----MOVE ROUTINE                                                        CSP10230
12    CALL MOVE(NCARD,N1,N2,NCARD,N3)                                     CSP10240
      GO TO 20                                                            CSP10250
C-----NZONE ROUTINE                                                       CSP10260
13    CALL NZONE(NCARD,N1,N2,N3)                                          CSP10270
      ANS=N3                                                              CSP10280
      GO TO 19                                                            CSP10290
C-----EDIT ROUTINE                                                        CSP10300
14    CALL EDIT(NCARD,N1,N2,NCARD,N3,N4)                                  CSP10310
      GO TO 20                                                            CSP10320
C-----GET ROUTINE                                                         CSP10330
15    ANS=GET(NCARD,N1,N2,V3)                                             CSP10340
      GO TO 19                                                            CSP10350
C-----PUT ROUTINE                                                         CSP10360
16    CALL PUT(NCARD,N1,N2,VAR,V3,N4)                                     CSP10370
```

```
      GO TO 20                                                          CSP10380
C-----FILL ROUTINE                                                      CSP10390
17    CALL FILL(NCARD,N1,N2,NVAR)                                       CSP1040D
      GO TO 20                                                          CSP10410
19    WRITE (NWRIT,8) ANS                                               CSP10420
20    WRITE (NWRIT,5) NCARD                                             CSP10430
      GO TO 10                                                          CSP10440
22    WRITE(NWRIT,7) NCARD                                              CSP10450
C-----A1DEC ROUTINE                                                     CSP10460
      CALL A1DEC(NCARD,N1,N2,NER1)                                      CSP10470
      CALL A1DEC(NCARD,N3,N4,NER2)                                      CSP10480
      N=N-7                                                             CSP10490
      GO TO (23,24,25,26,27,28),N                                       CSP10500
C-----ADD ROUTINE                                                       CSP10510
23    CALL ADD(NCARD,N1,N2,NCARD,N3,N4,NER3)                            CSP10520
      GO TO 29                                                          CSP10530
C-----SUB ROUTINE                                                       CSP10540
24    CALL SUB(NCARD,N1,N2,NCARD,N3,N4,NER3)                            CSP10550
      GO TO 29                                                          CSP10560
C-----MPY ROUTINE                                                       CSP10570
25    CALL MPY(NCARD,N1,N2,NCARD,N3,N4,NER3)                            CSP10580
      GO TO 29                                                          CSP10590
C-----DIV ROUTINE                                                       CSP10600
26    CALL DIV(NCARD,N1,N2,NCARD,N3,N4,NER3)                            CSP10610
      GO TO 29                                                          CSP10620
C-----ICOMP ROUTINE                                                     CSP10630
27    NER3=ICOMP(NCARD,N1,N2,NCARD,N3,N4)                               CSP10640
      GO TO 29                                                          CSP10650
C-----NSIGN ROUTINE                                                     CSP10660
28    CALL NSIGN(NCARD,N1,NVAR,NER3)                                    CSP10670
C-----DECA1 ROUTINE                                                     CSP10680
29    CALL DECA1(NCARD,N1,N2,NER4)                                      CSP10690
      IF(N-3)  33,32,30                                                 CSP10700
30    IF(N-4)  33,31,33                                                 CSP10710
31    JSPAN=N2-N1                                                       CSP10720
      KSPAN=N4-N3                                                       CSP10730
      KSTRT=N3-JSPAN-1                                                  CSP10740
      N3=N4-JSPAN                                                       CSP10750
      CALL DECA1(NCARD,KSTRT,N3-1,NER5)                                 CSP10760
      GO TO 33                                                          CSP10770
32    N3=N3-N2+N1-1                                                     CSP10780
33    CALL DECA1(NCARD,N3,N4,NER5)                                      CSP10790
      WRITE(NWRIT,6) NER1,NER2,NER3,NER4,NER5,NCARD                     CSP10800
      GO TO 10                                                          CSP10810
      END                                                               CSP10820
```



```
VARIABLE ALLOCATIONS
 V1   =0000  V2   =0003  V3   =0006  V4   =0009  VAR  =000C  ANS  =000F  NCARD=0064  NAMES=00A5  N    =00A6  NREAD=00A7
 NWRIT=00A8  I    =00A9  N1   =00AA  N2   =00AB  N3   =00AC  N4   =00AD  NVAR =00AE  NER1 =00AF  NER2 =00B0  NER3 =00B1
 NER4 =00B2  NER5 =00B3  JSPAN=00B4  KSPAN=00B5  KSTRT=00B6

STATEMENT ALLOCATIONS
 1    =00C0  2    =00C3  3    =00C8  4    =00E7  5    =00F2  6    =00FD  7    =010D  8    =0122  10   =0157  98   =016A
 99   =016C  21   =01C8  11   =01DA  12   =01E6  13   =01EF  14   =01FC  15   =0206  16   =0210  17   =021A  19   =0222
 20   =0228  22   =0231  23   =0254  24   =025F  25   =026A  26   =0275  27   =0280  28   =028C  29   =0292  30   =02A0
 31   =02A6  32   =02CE  33   =02D8

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CALLED SUBPROGRAMS
 DATSW   NCOMP   MOVE    NZONE   EDIT    GET     PUT     FILL    A1DEC   ADD     SUB     MPY     DIV     ICOMP   NSIGN
 DECA1   ELD     ESTO    IFIX    FLOAT   WRTYZ   SRED    SWRT    SCOMP   SF10    SIOAI   SIOIX   SIOF    SIO1    SUBSC
 STOP    CARDZ   PRNT2

INTEGER CONSTANTS
     0=00B8      2=00B9      1=008A      5=00BB      7=00BC      3=00BD      4=00BE      0=008F

CORE REQUIREMENTS FOR SMPL1
 COMMON       0  VARIABLES    184  PROGRAM    570

END OF COMPILATION
```

Sample Problem 1: Output

```
// XEQ                                                          CSP10830

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS   1.00000  10.00000  11.00000   0.00000   0.000
CARD BEFORE=ABCDEFGHIJKLMNOPQRST                                                    2CSP10860
ANSWER IS          -544.000
CARD AFTER =ABCDEFGHIJKLMNOPQRST                                                    2CSP10860

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS   1.00000  10.00000  11.00000   0.00000   0.000
CARD BEFORE=BC80 F    BC80 F                                                        4CSP10880
ANSWER IS             0.000
CARD AFTER =BC80 F    BC80 F                                                        4CSP10880

NOW TESTING 1130 CSP ROUTINE NCOMP WITH PARAMETERS  20.00000  25.00000  30.00000   0.00000   0.000
CARD BEFORE=                JKLMN     CBAFG                                       6CSP10900
ANSWER IS          448.000
CARD AFTER =                JKLMN     CBAFG                                       6CSP10900

NOW TESTING 1130 CSP ROUTINE MOVE  WITH PARAMETERS   1.00000   5.00000  20.00000   0.00000   0.000
CARD BEFORE=ABCDE                                                                   8CSP10920
CARD AFTER =ABCDE           ABCDE                                                   8CSP10920

NOW TESTING 1130 CSP ROUTINE MOVE  WITH PARAMETERS  40.00000  49.00000   1.00000   0.00000   0.000
CARD BEFORE=                                    9876543210                          10CSP10940
CARD AFTER =9876543210                          9876543210                          10CSP10940

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=          A                                                             12CSP10960
ANSWER IS             1.000
CARD AFTER =          A                                                             12CSP10960

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=          1                                                             14CSP10980
ANSWER IS             1.000
CARD AFTER =          I                                                             14CSP10980

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=                0                                                       16CSP11000
ANSWER IS             4.000
CARD AFTER =                0                                                       16CSP11000

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=                9                                                       18CSP11020
ANSWER IS             4.000
CARD AFTER =                9                                                       18CSP11020

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=                              J                                         20CSP11040
ANSWER IS             2.000
CARD AFTER =                              J                                         20CSP11040

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   5.00000   0.00000   0.00000   0.000
CARD BEFORE=                              R                                         22CSP11060
ANSWER IS             2.000
```

```
CARO AFTER =                              R                                                    22CSP11060

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000   0.000
CARO BEFORE=          A                                                                        24CSP11080
ANSWER IS               1.000
CARO AFTER =          A                                                                        24CSP11080

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000   0.000
CARO BEFORE=          1                                                                        26CSP11100
ANSWER IS               4.000
CARO AFTER =          A                                                                        26CSP11100

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  10.00000   1.00000   0.00000   0.00000   0.000
CARO BEFORE=          J                                                                        28CSP11120
ANSWER IS               2.000
CARO AFTER =          A                                                                        28CSP11120

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   4.00000   0.00000   0.00000   0.000
CARO BEFORE=                I                                                                  30CSP11140
ANSWER IS               1.000
CARO AFTER =                9                                                                30CSP11140

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   2.00000   0.00000   0.00000   0.000
CARO BEFORE=                9                                                                  32CSP11160
ANSWER IS               4.000
CARO AFTER =                R                                                                32CSP11160

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  20.00000   3.00000   0.00000   0.00000   0.000
CARO BEFORE=                R                                                                  34CSP11180
ANSWER IS               2.000
CARO AFTER =                Z                                                                34CSP11180

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   3.00000   0.00000   0.00000   0.000
CARO BEFORE=                            0                                                      36CSP11200
ANSWER IS               1.000
CARO AFTER =                            0                                                      36CSP11200

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   2.00000   0.00000   0.00000   0.000
CARO BEFORE=                            4                                                      38CSP11220
ANSWER IS               4.000
CARO AFTER =                            M                                                      38CSP11220

NOW TESTING 1130 CSP ROUTINE NZONE WITH PARAMETERS  30.00000   4.00000   0.00000   0.00000   0.000
CARO BEFORE=                            M                                                      40CSP11240
ANSWER IS               2.000
CARO AFTER =                            4                                                      40CSP11240

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   6.00000  20.00000  30.00000   0.000
CARO BEFORE=123456                  ,  $.  CR                                                  42CSP11260
CARO AFTER =123456                $1.234.56                                                    42CSP11260

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   6.00000  20.00000  30.00000   0.000
CARO BEFORE=02343K                  .  $.  CR                                                  44CSP11280
CARO AFTER =02343K                 $234.32CR                                                   44CSP11280

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   6.00000  20.00000  29.00000   0.000

CARO BEFORE=00343-                  ,  $.  -                                                   46CSP11300
CARO AFTER =00343-                   $34.30-                                                   46CSP11300

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   7.00000  21.00000  28.00000   0.000
CARO BEFORE=1234567                 ,  $.                                                      48CSP11320
CARO AFTER =1234567               ********                                                     48CSP11320

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   6.00000  10.00000  30.00000   0.000
CARO BEFORE=00005M                 ,*  .  CR                                                   50CSP11340
CARO AFTER =00005M   **************00.54CR                                                    50CSP11340

NOW TESTING 1130 CSP ROUTINE EOIT  WITH PARAMETERS   1.00000   6.00000  20.00000  29.00000   0.000
CARO BEFORE=    5M                 ,0  .  -                                                   52CSP11360
CARO AFTER =    5M                    .54-                                                    52CSP11360

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   0.01000   0.00000   0.000
CARO BEFORE=12345                                                                            54CSP11380
ANSWER IS             123.449
CARO AFTER =12345                                                                            54CSP11380

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   0.01000   0.00000   0.000
CARO BEFORE=1234N                                                                            56CSP11400
ANSWER IS            -123.449
CARO AFTER =1234N                                                                            56CSP11400

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   7.00000   0.00100   0.00000   0.000
CARO BEFORE=1 3 5 7                                                                          58CSP11420
ANSWER IS           1030.506
CARO AFTER =1 3 5 7                                                                          58CSP11420

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   1.00000   0.00000   0.000
CARO BEFORE=12A84                                                                            60CSP11440
ANSWER IS               0.000
CARO AFTER =12A84                                                                            60CSP11440

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   5.00000   1.00000   0.00000   0.000
CARO BEFORE=1230-                                                                            62CSP11460
ANSWER IS          -12300.000
CARO AFTER =1230-                                                                            62CSP11460

NOW TESTING 1130 CSP ROUTINE GET   WITH PARAMETERS   1.00000   3.00000   0.00001   0.00000   0.000
CARO BEFORE=123                                                                              64CSP11480
ANSWER IS               0.001
CARO AFTER =123                                                                              64CSP11480

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS   1.00000   5.00000   0.50000   0.00000 12345.000
CARO BEFORE=                                                                                 66CSP11500
CARO AFTER =12345                                                                            66CSP11500

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS   1.00000   2.00000   5.00000   1.00000 12890.000
CARO BEFORE=                                                                                 68CSP11520
CARO AFTER =89                                                                               68CSP11520

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  11.00000  15.00000   5.00000   1.00000 12345.000
```

```
CARD BEFORE=                                                                                                    70CSP11540
CARD AFTER =          01235                                                                                     70CSP11540

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  10.00000  16.00000  50.00000   2.00000-34567.000
CARD BEFORE=                                                                                                    72CSP11560
CARD AFTER =          00D034D                                                                                   72CSP11560

NOW TESTING 1130 CSP ROUTINE PUT   WITH PARAMETERS  10.00000  17.00000   5.00000   1.00000   -16.000
CARD BEFORE=                                                                                                    74CSP11580
CARD AFTER =          0000000K                                                                                  74CSP11580

NOW TESTING 1130 CSP ROUTINE FILL  WITH PARAMETERS   1.00000  10.00000   0.00000   0.00000 16448.000
CARD BEFORE=ABCDEFGHIJK                                                                                         76CSP11600
CARD AFTER =           K                                                                                        76CSP11600

NOW TESTING 1130 CSP ROUTINE FILL  WITH PARAMETERS  20.00000  25.00000   0.00000   0.00000 23360.000
CARD BEFORE=                ABCDEFGH                                                                            78CSP11620
CARD AFTER =                A$$$$$$H                                                                            78CSP11620

NOW TESTING 1130 CSP ROUTINE AOD   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.0D0

   INDICATORS   CARD BEFORE=                                  24                                    2048  CSP11640
 0  0  0  0  0  CARD AFTER =                                 00024                                  02072  CSP11640

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  24                                    2048  CSP11660
 0  0  0  0  0  CARD AFTER =                                 00024                                  02024  CSP11660

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  24                                    2048  CSP11680
 0  0  0  0  0  CARD AFTER =                                 00024                            0000049152  CSP11680

NOW TESTING 1130 CSP ROUTINE OIV   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  24                                    2048  CSP11700
 0  0  0  0  0  CARD AFTER =                                 00024                            0008500008  CSP11700

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  24                                    2048  CSP11720
 0  0 -2  0  0  CARD AFTER =                                 00024                                  02048  CSP11720

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000      1.000

   INDICATORS   CARD BEFORE= 65                                                                             CSP11740
 0  0  1  0  0  CARD AFTER = 65                                                                             CSP11740

NOW TESTING 1130 CSP ROUTINE AOD   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  99                                    2048  CSP11760
 0  0  0  0  0  CARD AFTER =                                 D0099                                  02147  CSP11760

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  99                                    2048  CSP11780
 0  0  0  0  0  CARD AFTER =                                 00099                                  01949  CSP11780

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  99                                    2048  CSP11800
 0  0  0  0  0  CARD AFTER =                                 00099                            0000202752  CSP11800

NOW TESTING 1130 CSP ROUTINE OIV   WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  99                                    2048  CSP11820
 0  0  0  0  0  CARD AFTER =                                 00099                                0002000068  CSP11820

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  31.00000  35.00000  66.00000  70.00000      0.000

   INDICATORS   CARD BEFORE=                                  99                                    2048  CSP11840
 0  0 -2  0  0  CARD AFTER =                                 00099                                  02048  CSP11840

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000     -1.000

   INDICATORS   CARD BEFORE= 54                                                                             CSP11860
 0  0  1  0  0  CARD AFTER = N4                                                                             CSP11860

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

   INDICATORS   CARD BEFORE= 12345678901234567890  1234567890123456789012345678901234567890  CSP11880
 0  0  0  0  0  CARD AFTER = 12345678901234567890  1234567890246913578024691357 80           CSP11880

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

   INDICATORS   CARD BEFORE= 12345678901234567890  1234567890123456789012345678901234567890  CSP11900
 0  0  0  0  0  CARD AFTER = 12345678901234567890  12345678900000000000000000000000000000000  CSP11900

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

   INDICATORS   CARD BEFORE= 12345678901234567890  1234567890123456789012345678901234567890  CSP11920
 0  0  0  0  0  CARD AFTER = 12345678901234567890015241578753238836750342935775019051998750 52100  CSP11920

NOW TESTING 1130 CSP ROUTINE OIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

   INDICATORS   CARD BEFORE= 12345678901234567890  1234567890123456789012345678901234567890  CSP11940
 0  0  0  0  0  CARD AFTER = 12345678901234567890000000000000000000001000000000000000000001234567890  CSP11940

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000

   INDICATORS   CARD BEFORE= 12345678901234567890  1234567890123456789012345678901234567890  CSP11960
 0  0 -1  0  0  CARD AFTER = 12345678901234567890  1234567890123456789012345678901234567890  CSP11960

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000      0.000

   INDICATORS   CARD BEFORE= 32                                                                             CSP11980
 0  0  1  0  0  CARD AFTER = L2                                                                             CSP11980

NOW TESTING 1130 CSP ROUTINE AOD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000      0.000
```

```
    INDICATORS   CARD BEFORE= 1234567890123456789-                    12345678901234567890123456789D  CSP12000
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                    12345678900000000000000000000-  CSP12000

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                    123456789D123456789012345678 9D  CSP12020
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                    123456789D2469135780246913578 0  CSP12020

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                    12345678901234567890123456789D  CSP1204D
  0  0  0  0  0  CARD AFTER = 1234567890123456789-01524157875323883675034293577501905199875019052 10-  CSP1204D

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                    12345678901234567890123456789D  CSP12060
  0  0  0  0  0  CARD AFTER = 1234567890123456789-00000000000000000001000000000-0000000000123456789D  CSP12D60

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                    12345678901234567890123456789D  CSP12080
  0  0 -1  0  0  CARD AFTER = 1234567890123456789-                    12345678901234567890123456789D  CSP12080

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000     1.000

    INDICATORS   CARD BEFORE= ON                                                                     CSP12100
  0  0 -1  0  0  CARD AFTER = 6N                                                                     CSP12100

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   12345678901234567890123456789-  CSP12120
  0  0  0  0  0  CARD AFTER = 12345678901234567890                   12345678900000000000000000000-  CSP12120

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   12345678901234567890123456789-  CSP12140
  0  0  0  0  0  CARD AFTER = 12345678901234567890                   12345678902469135780246913578-  CSP12140

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   12345678901234567890123456789-  CSP12160
  0  0  0  0  0  CARD AFTER = 12345678901234567890D1524157875323883675034293577501905199875019052 10-  CSP12160

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   12345678901234567890123456789-  CSP12180
  0  0  0  0  0  CARD AFTER = 12345678901234567890000000000000000000001000000000-0000000000123456789-  CSP12180

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   12345678901234567890123456789-  CSP12200
  0  0  1  0  0  CARD AFTER = 12345678901234567890                   12345678901234567890123456789-  CSP12200

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000    -1.000

    INDICATORS   CARD BEFORE= NM                                                                     CSP12220


  0  0 -1  0  0  CARD AFTER = NM                                                                     CSP12220

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                   12345678901234567890123456789-  CSP12240
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                   12345678902469135780246913578-  CSP12240

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                   12345678901234567890123456789-  CSP12260
  0  0  0  0  0  CARD AFTER = 1234567890123456789-                   12345678900000000000000000000-  CSP12260

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                   12345678901234567890123456789-  CSP12280
  0  0  0  0  0  CARD AFTER = 1234567890123456789-01524157875323883675034293577501905199875019052100  CSP12280

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                   12345678901234567890123456789-  CSP12300
  0  0  0  0  0  CARD AFTER = 1234567890123456789-00000000000000000001000000000000000000000123456789-  CSP12300

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  41.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 1234567890123456789-                   12345678901234567890123456789-  CSP12320
  0  0  1  0  0  CARD AFTER = 1234567890123456789-                   12345678901234567890123456789-  CSP12320

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS   1.00000   1.00000   2.00000   2.00000     0.000

    INDICATORS   CARD BEFORE= ML                                                                     CSP12340
  0  0 -1  0  0  CARD AFTER = 4L                                                                     CSP12340

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                  12345678901234567890  CSP12360
  0  0  0  0  0  CARD AFTER = 12345678901234567890                                  24691357802469135780  CSP12360

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                  12345678901234567890  CSP12380
  0  0  0  0  0  CARD AFTER = 12345678901234567890                                  00000000000000000000  CSP12380

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                  12345678901234567890  CSP12400
  0  0  0  0  0  CARD AFTER = 12345678901234567890          01524157875323883675019051998750190521 00  CSP12400

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                  12345678901234567890  CSP12420
  0  0  0  0  0  CARD AFTER = 12345678901234567890          00000000000000000001000000000000000000000  CSP12420

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS   1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                                  12345678901234567890  CSP12440
```

```
 0  0  0  0  0  CARD AFTER = 12345678901234567890                                  12345678901234567890  CSP12440

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS  1.00000   1.00000   2.00000   2.00000     1.000

    INDICATORS   CARD BEFORE= -0                                                                      CSP12460
 0  0 -1  0  0   CARD AFTER = 00                                                                      CSP12460

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                  12345678901234567890  CSP12480
 0  0  0  0  0   CARD AFTER = 12345678901234567890-                  00000000000000000000  CSP12480

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                  12345678901234567890  CSP12500
 0  0  0  0  0   CARD AFTER = 12345678901234567890-                  24691357802469135780  CSP12500

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                  12345678901234567890  CSP12520
 0  0  0  0  0   CARD AFTER = 12345678901234567890-  01524157875323883675019051998750190521O-  CSP12520

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                  12345678901234567890  CSP12540
 0  0  0  0  0   CARD AFTER = 12345678901234567890-  00000000000000000000J00000000000000000000  CSP12540

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890-                  12345678901234567B90  CSP12560
 0  0 -1  0  0   CARD AFTER = 12345678901234567890-                  12345678901234567890  CSP12560

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS  1.00000   1.00000   2.00000   2.00000    -1.000

    INDICATORS   CARD BEFORE= -0                                                                      CSP12580
 0  0 -1  0  0   CARD AFTER = -0                                                                      CSP12580

NOW TESTING 1130 CSP ROUTINE ADD   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   1234567890123456789-  CSP12600
 0  0  0  0  0   CARD AFTER = 12345678901234567890                   00000000000000000000-  CSP12600

NOW TESTING 1130 CSP ROUTINE SUB   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   1234567890123456789-  CSP12620
 0  0  0  0  0   CARD AFTER = 12345678901234567890                   2469135780246913578-  CSP12620

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567890                   1234567890123456789-  CSP12640
 0  0  0  0  0   CARD AFTER = 12345678901234567890  01524157875323BB3675019051998750190521O-  CSP12640

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567B90                   1234567B90123456789-  CSP12660
```

```
 0  0  0  0  0  CARD AFTER = 12345678901234567890  00000000000000000000J00000000000000000000-  CSP1266D

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 12345678901234567B90                   1234567B90123456789-  CSP12680
 0  0  1  0  0   CARD AFTER = 12345678901234567B90                   1234567B90123456789-  CSP12680

NOW TESTING 1130 CSP ROUTINE NSIGN WITH PARAMETERS  1.00000   1.00000   2.00000   2.00000     0.000

    INDICATORS   CARD BEFORE= -0                                                                      CSP12700
 0  0 -1  0  0   CARD AFTER = 00                                                                      CSP12700

NOW TESTING 1130 CSP RDUTINE ADD   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 123456789012345679-                    1234567890123456789-  CSP12720
 0  0  0  0  0   CARD AFTER = 123456789012345679-                    2469135780246913578-  CSP12720

NOW TESTING 1130 CSP RDUTINE SUB   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 123456789012345679-                    1234567890123456789-  CSP12740
 0  0  0  0  0   CARD AFTER = 123456789012345679-                    00000000000000000000-  CSP12740

NOW TESTING 1130 CSP ROUTINE MPY   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFDRE= 123456789012345679-                    1234567890123456789-  CSP12760
 0  0  0  0  0   CARD AFTER = 123456789012345679-  01524157875323883675019051998750190521O0  CSP12760

NOW TESTING 1130 CSP ROUTINE DIV   WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFDRE= 123456789012345679-                    1234567890123456789-  CSP12780
 0  0  0  0  0   CARD AFTER = 123456789012345679-  000000D00000000000010000000000000000000-  CSP12780

NOW TESTING 1130 CSP ROUTINE ICOMP WITH PARAMETERS  1.00000  20.00000  51.00000  70.00000     0.000

    INDICATORS   CARD BEFORE= 123456789012345679-                    1234567890123456789-  CSP12800
 0  0  0  0  0   CARD AFTER = 123456789012345679-                    1234567890123456789-  CSP12800
```

Sample Problem 1: Data Input Listing

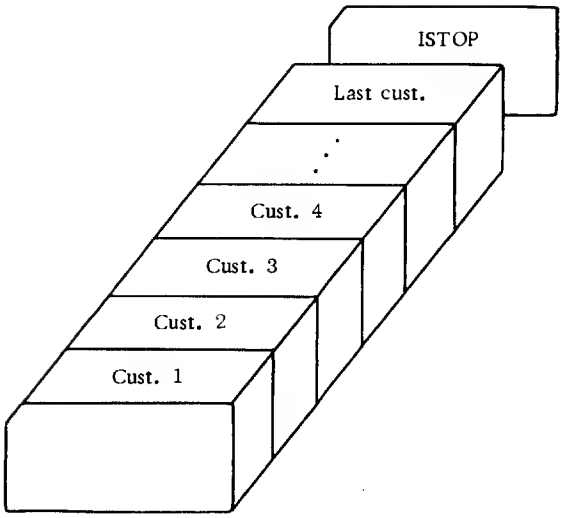
```
NCOMPMOVE NZONEEDIT GET  PUT  FILL ADD  SUB  MPY  DIV  ICOMPNSIGN      CSP10840
         1         1         10        11                             1CSP10850
ABCDEFGHIJKLMNOPQRST
         1         1         10        11                             2CSP10860
                                                                       3CSP10870
BCBD F    BCBD F
         1        20         25        30                             4CSP10880
                                                                       5CSP10890
                   JKLMN      CBAFG                                    6CSP10900
         2         1          5        20                             7CSP10910
ABCDE
                                                                       8CSP10920
         2        40         49         1                             9CSP10930
                                        9876543210                     10CSP10940
         3        10          5                                       11CSP10950
         A
         3        10          5                                       12CSP10960
                                                                       13CSP10970
         I
         3        20          5                                       14CSP10980
                                                                       15CSP10990
          0
         3        20          5                                       16CSP11000
                                                                       17CSP11010
          9
         3        30          5                                       18CSP11020
                              J                                        19CSP11030
         3        30          5                                       20CSP11040
                              R                                        21CSP11050
         3        10          1                                       22CSP11060
         A
         3        10          1                                       23CSP11070
                                                                       24CSP11080
         1
         3        10          1                                       25CSP11090
                                                                       26CSP11100
         J
         3        20          4                                       27CSP11110
                                                                       28CSP11120
          1
         3        20          2                                       29CSP11130
                                                                       30CSP11140
          9
         3        20          3                                       31CSP11150
                                                                       32CSP11160
          R
         3        30          3                                       33CSP11170
                              D                                        34CSP11180
         3        30          2                                       35CSP11190
                              4                                        36CSP11200
         3        30          4                                       37CSP11210
                              M                                        38CSP11220
         4         1          6        20        30                   39CSP11230
                                                                       40CSP11240
123456             .  $.  CR                                          41CSP11250
         4         1          6        20        30                   42CSP11260
02343K             .  $.  CR                                          43CSP11270
         4         1          6        20        29                   44CSP11280
00343-             .  $.  -                                           45CSP11290
         4         1          7        21        28                   46CSP11300
1234567            .  $.                                              47CSP11310
         4         1          6        10        30                   48CSP11320
00005M               *  .  CR                                         49CSP11330
         4         1          6        20        29                   50CSP11340
    5M               0  .  -                                         51CSP11350
                                                                       52CSP11360
```

```
         5          1          5        .01                                                  53CSP11370
12345                                                                                       54CSP11380
         5          1          5        .01                                                  55CSP11390
1234N                                                                                       56CSP11400
         5          1          7       .001                                                  57CSP11410
1 3 5 7                                                                                     58CSP11420
         5          1          5         1.                                                  59CSP11430
12A84                                                                                       60CSP11440
         5          1          5         1.                                                  61CSP11450
1230-                                                                                       62CSP11460
         5          1          3     .00001                                                  63CSP11470
123                                                                                         64CSP11480
         6          1          5        0.5          0     12345.                          65CSP11490
                                                                                            66CSP11500
         6          1          2        5.0          1     12890.                          67CSP11510
                                                                                            68CSP11520
         6         11         15        5.0          1     12345.                          69CSP11530
                                                                                            70CSP11540
         6         10         16       50.0          2    -34567.                          71CSP11550
                                                                                            72CSP11560
         6         10         17        5.0          1       -16.                          73CSP11570
                                                                                            74CSP11580
         7          1         10                                16448.                       75CSP11590
ABCDEFGHIJK                                                                                 76CSP11600
         7         20         25                                23360.                       77CSP11610
                   A8C0EFGH                                                                 78CSP11620
        08         31         35          66         70                                      CSP11630
                                  24                                          2048          CSP11640
        09         31         35          66         70                                      CSP11650
                                  24                                          2048          CSP11660
        10         31         35          66         70                                      CSP11670
                                  24                                          2048          CSP11680
        11         31         35          66         70                                      CSP11690
                                  24                                          2048          CSP11700
        12         31         35          66         70                                      CSP11710
                                  24                                          2048          CSP11720
        13          1          1           2          2         1.                          CSP11730
                                                                                            CSP11740
65                                                                                          CSP11750
        08         31         35          66         70                                      CSP11760
                                  99                                          2048          CSP11770
        09         31         35          66         70                                      CSP11780
                                  99                                          2048          CSP11790
        10         31         35          66         70                                      CSP11800
                                  99                                          2048          CSP11810
        11         31         35          66         70                                      CSP11820
                                  99                                          2048          CSP11830
        12         31         35          66         70                                      CSP11840
                                  99                                          2048          CSP11850
        13          1          1           2          2        -1.                          CSP11860
                                                                                            CSP11870
54                                                                                          CSP11880
        08         01         20          41         70
12345678901234567890                    12345678901234567890                              CSP11890
        09         01         20          41         70
12345678901234567890                    12345678901234567890                              CSP11900
```

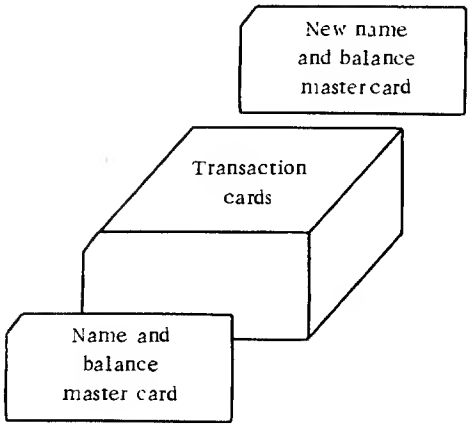
```
        10         01         20          41         70
12345678901234567890                    12345678901234567890                              CSP11910
        11         01         20          41         70                                     CSP11920
12345678901234567890                    12345678901234567890                              CSP11930
        12         01         20          41         70                                     CSP11940
12345678901234567890                    12345678901234567890                              CSP11950
        13          1          1           2          2                                     CSP11960
                                                                                            CSP11970
32                                                                                          CSP11980
        08         01         20          41         70                                     CSP11990
12345678901234567890-                   12345678901234567890                              CSP12000
        09         01         20          41         70                                     CSP12010
12345678901234567890-                   12345678901234567890                              CSP12020
        10         01         20          41         70                                     CSP12030
12345678901234567890-                   12345678901234567890                              CSP12040
        11         01         20          41         70                                     CSP12050
12345678901234567890-                   12345678901234567890                              CSP12060
        12         01         20          41         70                                     CSP12070
12345678901234567890-                   12345678901234567890                              CSP12080
        13          1          1           2          2         1.                          CSP12090
                                                                                            CSP12100
ON                                                                                          CSP12110
        08         01         20          41         70
12345678901234567890                    12345678901234567890-                             CSP12120
        09         01         20          41         70                                     CSP12130
12345678901234567890                    12345678901234567890-                             CSP12140
        10         01         20          41         70                                     CSP12150
12345678901234567890                    12345678901234567890-                             CSP12160
        11         01         20          41         70                                     CSP12170
12345678901234567890                    12345678901234567890-                             CSP12180
        12         01         20          41         70                                     CSP12190
12345678901234567890                    12345678901234567890-                             CSP12200
        13          1          1           2          2        -1.                          CSP12210
                                                                                            CSP12220
NM                                                                                          CSP12230
        08         01         20          41         70
12345678901234567890-                   12345678901234567890-                             CSP12240
        09         01         20          41         70                                     CSP12250
12345678901234567890-                   12345678901234567890-                             CSP12260
        10         01         20          41         70                                     CSP12270
12345678901234567890-                   12345678901234567890-                             CSP12280
        11         01         20          41         70                                     CSP12290
12345678901234567890-                   12345678901234567890-                             CSP12300
```

PROBLEM 2

The purpose of this program is to create invoices. The input deck is as follows:



Input deck



Detailed description of individual customer deck

Each customer has the old master name and balance card, followed by the transaction cards, followed by a blank master name and balance card.

The invoice is printed as in the example, and a new master name and balance card image is printed on the console printer. Then the next customer is processed until the stop code card is reached (ISTOP in cc 1-5).

In an actual situation the new card image would be punched and stacker-selected. Then, as input to the next run of the program, a new input deck would have to be prepared.

This problem requires an 1132 Printer on the system.

Sample Problem 2: Detailed Description

1. Read all constant information.

2. Initialize error indicators.

   a. J=2

   b. I=0, L=0, M=0

3. Read the first card. It should be a master card.

4. Is the card read in 3 the last card?

   No — 5 Yes — 64

5. Is the card read in 3 above a master card?

   No — 72 Yes — 6

6. Go to the top of a new page.

7. Clear the print area.

8. Print the customer name.

9. Move the edit mark to the work area.

10. Edit the previous balance.

11. Print the customer street address.

12. Move the words PREVIOUS BALANCE to the print area.

13. Move the work area to the print area.

14. Print the customer city, state, and zip code.

15. Skip 3 lines.

16. Print the column headings.

17. Print the print area.

18. Clear the print area.

19. Convert the previous balance from A1 format to decimal format.

20. Is the conversion in 19 correct?

    No — 66 Yes — 21

21. Set the total (ISUM) equal to the previous balance.

22. Set up the output area for the new master card.

23. Read a card.

24. Is the card read at 23 the last card?

    No — 25 Yes — 64

25. Is the card read at 23 a master card?

    No — 26 Yes — 52

26. Is the card read at 23 a transaction card?

    No — 49 Yes — 27

27. Is the card read at 23 for the same customer being processed?

    No — 49 Yes — 28

28. Move the item name to the print area.

29. Move the edit mask to the print area for dollar amount.

30. Move the edit mask to the print area for quantity.

31. Edit the quantity.

32. Edit the dollar amount.

33. Print the detail line assembled in 28 through 32.

34. Has channel 12 on the carriage tape been encountered?

    No — 35 Yes — 46

35. Convert the dollar amount from A1 format to decimal format.

36. Is the conversion in 35 correct?

    No — 40 Yes — 37

37. Add the dollar amount to ISUM.

38. Did overflow occur in the addition in 37?

    No — 23 Yes — 39

39. STOP and display 777.

40. Make the character in error a digit.

41. Try to convert only the character in error.

42. Is the conversion in 41 correct?

    No — 43 Yes — 44

43. STOP and display 666.

44. Convert the entire field back to A1 format.

45. Go to 35.

46. Go to the top of a new page.

47. Print the headings.

48. Go to 35.

49. Type ERROR on the console printer.

50. Type the card read on the console printer.

51. Go to 23.

52. Convert the total (ISUM) from decimal format to A1 format.

53. Is the conversion in 52 correct?

    No — 54 Yes — 55

54. STOP and display 555.

55. Clear the print area.

56. Move the edit mask to the print area.

57. Edit the total (ISUM).

58. Place the unedited total (ISUM) in the new master card.

59. Type the new master card image on the console printer.

60. Move the word TOTAL to the print area.

61. Skip 2 lines.

62. Print the print area, the total line.

63. Go to 2b.

64. Type END OF JOB.

65. STOP and display 111.

66. Make the character in error a digit.

67. Try to convert only the character in error.

68. Is the conversion in 67 correct?

    No — 69 Yes — 70

69. STOP and display 444.

70. Convert the entire field back to A1 format.

71. Go to 19.

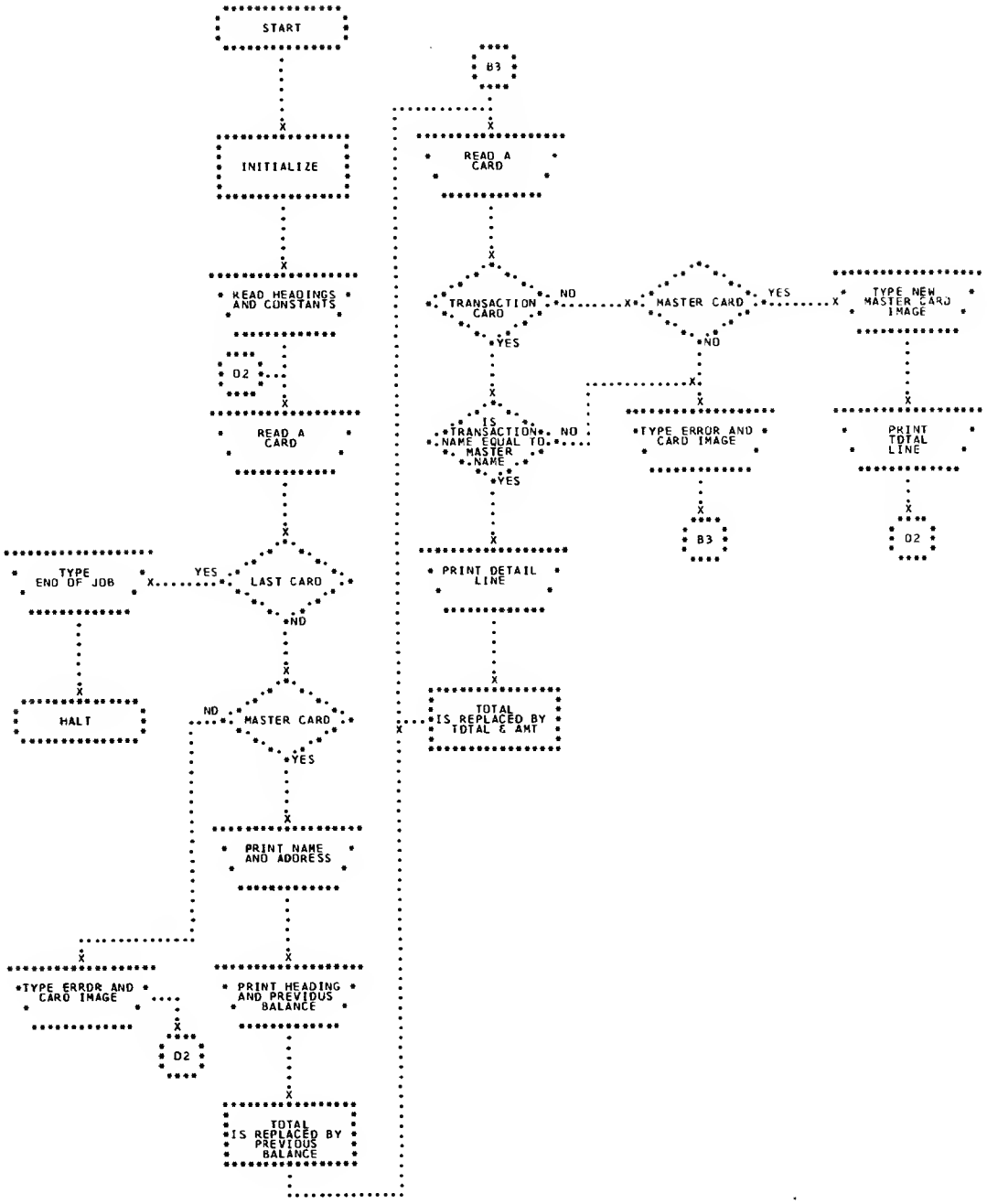
72. Type ERROR on the console printer.

73. Type the card read on the console printer.

74. Go to 2b.

Card Formats

1	Master	Customer Name	Street Address	City State Zone	Balance	Blank	A	Blank	CSP	Card Seq. No.
		1–20	21–40	41–60	61–68	69	70	71–72	73–75	76–80
2	Trans.	Customer Name	Item Name	Total Amt.	Qty	Blank	J	Blank	CSP	Card Seq. No.
		1–20	21–40	41–48	49–52	53–69	70	71–72	73–75	76–80


START
INITIALIZE
READ HEADINGS AND CONSTANTS
D2
READ A CARD
TYPE END OF JOB
YES
LAST CARD
NO
HALT
NO
MASTER CARD
YES
PRINT NAME AND ADDRESS
TYPE ERROR AND CARD IMAGE
D2
PRINT HEADING AND PREVIOUS BALANCE
TOTAL IS REPLACED BY PREVIOUS BALANCE
B3
READ A CARD
TRANSACTION CARD
NO
MASTER CARD
YES
TYPE NEW MASTER CARD IMAGE
YES
IS TRANSACTION NAME EQUAL TO MASTER NAME
NO
TYPE ERROR AND CARD IMAGE
B3
PRINT TOTAL LINE
D2
PRINT DETAIL LINE
TOTAL IS REPLACED BY TOTAL + AMT

Sample Problem 2: Source Program

```
// FOR                                                                  CSP1282D


                                                                        PAGE 01
** SAMPLE PROBLEM 2                                                    CSP1283D
* NAME SMPL2                                                           CSP1284D
* ONE WORD INTEGERS                                                    CSP1285D
* EXTENDED PRECISION                                                   CSP1286D
* LIST ALL                                                             CSP1287D


 SAMPLE PROBLEM 2                                                       PAGE 02
C-----THE INPUT IS MADE UP OF A MASTER CARD FOLLOWED BY THE TRANSACTION CSP1288D
C-----CARDS FOR EACH CUSTOMER. WE WANT TO PRINT AN INVOICE AND PRINT A  CSP1289D
C-----NEW MASTER CARD FOR EACH CUSTOMER.                                CSP1290D
      DIMENSION INCRD(82),IMASK(13),IPRNT(79),IOTCD(8D),ISTOP(5),       CSP1291D
     1IHEAD(80), IPRV8(16),ITOT(5),IWK(13),ISUM(8),IERDR(6),IEOJ(1D)    CSP1292D
      CALL READ(IEOJ,1,1D,J)                                            CSP1293D
      CALL READ(IERDR,1,6,J)                                            CSP1294D
      CALL READ(IMASK,1,13,J)                                           CSP1295D
      CALL READ(IPRV8,1,16,J)                                           CSP1296D
      CALL READ(IHEAD,1,72,J)                                           CSP1297D
      CALL READ(IHEAD,73,8D,J)                                          CSP1298D
      CALL READ(ISTOP,1,5,J)                                            CSP1299D
      CALL READ(ITOT,1,5,J)                                             CSP13D0D
      J=2                                                               CSP13D1D
      INCRD(81)=16448                                                   CSP13D2D
      INCRD(82)=544D                                                    CSP13D3D
1     I=D                                                               CSP13D4D
      L=D                                                               CSP13D5D
      M=D                                                               CSP13D6D
      CALL READ(INCRD,1,8D,J)                                           CSP13D7D
      IF(J-1)  22,2,2                                                   CSP13D8D
2     IF(NCOMP(INCRD,1,5,ISTOP,1))   3,22,3                             CSP13D9D
3     CALL NZONE(INCRD,7D,5,K)                                          CSP131DD
      IF(K-1)   26,4,26                                                 CSP131ID
4     CALL SKIP(12544)                                                  CSP13120
      CALL FILL(IPRNT,1,79,16448)                                       CSP13130
      CALL PRINT(INCRD,1,2D,I)                                          CSP13140
      CALL MOVE(IMASK,1,13,IWK,1)                                       CSP1315D
      CALL EDIT(INCRD,61,68,IWK,1,13)                                   CSP13160
      CALL PRINT(INCRD,21,4D,I)                                         CSP13170
      CALL MOVE(IPRV8,1,16,IPRNT,23)                                    CSP13180
      CALL MOVE(IWK,1,13,IPRNT,67)                                      CSP13190
      CALL PRINT(INCRD,41,6D,I)                                         CSP13200
      CALL SKIP(16128)                                                  CSP1321D
      CALL PRINT(IHEAD,1,8D,I)                                          CSP13220
      CALL PRINT(IPRNT,1,79,1)                                          CSP13230
      CALL FILL(IPRNT,1,79,16448)                                       CSP13240
4D    CALL A1DEC(INCRD,61,68,L)                                         CSP13250
      IF(L)   5,5,23                                                    CSP13260
5     CALL MOVE(INCRD,61,68,ISUM,1)                                     CSP13270
      CALL MOVE(INCRD,1,8D,IOTCD,1)                                     CSP13280
6     CALL READ(INCRD,1,8D,J)                                           CSP13290
      IF(J-1)   22,7,7                                                  CSP1330D
7     CALL NZONE(INCRD,7D,5,K)                                          CSP13310
      IF(K-1)   18,19,8                                                 CSP13320
8     IF(K-2)   18,9,18                                                 CSP13330
9     IF(NCOMP(INCRD,1,2D,IOTCD,1))   18,10,18                          CSP1334D
1D    CALL MOVE(INCRD,21,4D,IPRNT,23)                                   CSP13350
      CALL MOVE(IMASK,1,13,IPRNT,67)                                    CSP13360
      CALL MOVE(IMASK,3,8,IPRNT,7)                                      CSP13370
      IPRNT(12)=-4032                                                   CSP1338D
      CALL EDIT(INCRD,49,52,IPRNT,7,12)                                 CSP1339D
```

```
SAMPLE PROBLEM 2                                                    PAGE 03

      CALL EOIT(INCRO,41,48,IPRNT,67,79)                            CSP13400
      CALL PRINT(IPRNT,1,79,1)                                      CSP13410
      IF(I-3)    11,11,17                                           CSP13420
11    CALL A1OEC(INCRO,41,48,L)                                     CSP13430
      IF(L)    12,12,14                                             CSP13440
12    CALL AOO(INCRO,41,48,ISUM,1,8,M)                              CSP13450
      IF(M)    13,6,13                                              CSP13460
13    CALL IONO                                                     CSP13470
      STOP 777                                                      CSP13480
14    CALL NZONE(INCRD,L,4,N1)                                      CSP13490
      N1=0                                                          CSP13500
      CALL A1OEC(INCRO,L,L,N1)                                      CSP13510
      IF(N1)    16,16,15                                            CSP13520
15    CALL IONO                                                     CSP13530
      STOP 666                                                      CSP13540
16    CALL OECA1(INCRO,41,48,L)                                     CSP13550
      L=0                                                           CSP13560
      GO TO 11                                                      CSP13570
17    CALL SKIP(12544)                                              CSP13580
      CALL PRINT(IHEAO,1,80,1)                                      CSP13590
      I=0                                                           CSP13600
      GO TO 11                                                      CSP13610
18    CALL TYPER(IEROR,1,5)                                         CSP13620
      CALL TYPER(INCRO,1,82)                                        CSP13630
      GO TO 6                                                       CSP13640
19    CALL OECA1(ISUM,1,8,L)                                        CSP13650
      IF(L)    20,21,20                                             CSP13660
20    CALL IONO                                                     CSP13670
      STOP 555                                                      CSP13680
21    CALL FILL(IPRNT,1,79,16448)                                   CSP13690
      CALL MOVE(1MASK,1,13,IPRNT,67)                                CSP13700
      CALL EOIT(ISUM,1,8,IPRNT,67,79)                               CSP13710
      CALL MOVE(ISUM,1,8,1OTCO,61)                                  CSP13720
      CALL TYPER(IOTCO,1,80)                                        CSP13730
      CALL MOVE(1TOT,1,5,IPRNT,23)                                  CSP13740
      CALL SKIP(15872)                                              CSP13750
      CALL PRINT(IPRNT,1,79,1)                                      CSP13760
      CALL TYPER(INCRO,81,82)                                       CSP13770
      GO TO 1                                                       CSP13780
22    CALL TYPER(IEOJ,1,10)                                         CSP13790
      CALL IONO                                                     CSP13800
      STOP 111                                                      CSP13810
23    CALL NZONE(INCRO,L,4,N1)                                      CSP13820
      N1=0                                                          CSP13830
      CALL A1OEC(INCRO,L,L,N1)                                      CSP13840
      IF(N1)    25,25,24                                            CSP13850
24    CALL IONO                                                     CSP13860
      STOP 444                                                      CSP13870
25    CALL DECA1(INCRO,61,68,L)                                     CSP13880
      L=0                                                           CSP13890
      GO TO 40                                                      CSP13900
26    CALL TYPER(IEROR,1,5)                                         CSP13910


SAMPLE PROBLEM 2                                                    PAGE 04

      CALL TYPER(INCRD,1,B2)                                        CSP13920
      GO TO 1                                                       CSP13930
      ENO                                                           CSP13940


SAMPLE PROBLEM 2                                                    PAGE 05

VARIABLE ALLOCATIONS
 INCRO=00S1   IMASK=005E   IPRNT=00A0   IOTCO=00F0   ISTOP=0102   IHEAO=0152   IPRVB=0162   1TOT =0167   IWK  =0174   ISUM =017C
 IEROR=0182   IEOJ =018C   J    =0180   I    =018E   L    =018F   M    =0190   K    =0191   N1   =0192

STATEMENT ALLOCATIONS
 1    =0206   2    =021E   3    =0227   4    =0233   40   =0280   5    =028A   6    =0298   7    =02A4   8    =02B2   9    =02B8
 10   =02C1   11   =02F9   12   =0303   13   =0310   14   =0314   15   =0328   16   =032C   17   =0338   18   =0347   19   =0353
 20   =0350   21   =0361   22   =0399   23   =03A2   24   =0386   25   =03BA   26   =03C6

FEATURES SUPPORTEO
 ONE WORD INTEGERS
 EXTENOEO PRECISION

CALLEO SUBPROGRAMS
 REAO    NCOMP   NZONE   SKIP    FILL    PRINT   MOVE    EOIT    A1OEC   AOO     IONO    OECA1   TYPER   STOP

INTEGER CONSTANTS
     1=0196      10=0197       6=0198      13=0199      16=019A      72=019B      73=019C      80=0190       5=019E       2=019F
 16448=01A0    5440=01A1       0=01A2      70=01A3   12544=01A4      79=01A5      20=01A6      61=01A7      68=01A8      21=01A9
    40=01AA      23=01AB      67=01AC      41=01A0      60=01AE   16128=01AF       3=01B0       8=01B1       7=01B2    4032=0183
    49=0184      52=01B5      12=0186      48=0187     777=0188       4=01B9     666=01BA      82=018B    555=01BC   15872=01B0
    81=018E     111=01BF    444=01C0    1911=01C1    1638=01C2    1365=01C3     273=01C4    1092=01C5

CORE REQUIREMENTS FOR SMPL2
 COMMON       0  VARIABLES     406  PROGRAM     572

ENO OF COMPILATION


// XEQ                                                              CSP13950
```

Sample Problem 2: Invoice Output

OAVES MARKET
1997 WASHIMGTON ST.
MEWTOWM. MASS. 02158

OTY	MAME	AMT
	PREVIOUS 8ALANCE	$111.29
8	SUGAR - BAGS	$21.02
11	CMICKEM SOUP - CASES	$38.76
10	TOMATO SOUP - CASES	$30.11
8	SUGAR RETURMEO	$21.02CR
6	COOKIES - CASES	$45.21
17	GINGER ALE - CASES	$52.37
17	ROOT BEER - CASES	$52.37
17	ORAMGE A0E - CASES	$52.37
17	CREME S00A - CASES	$52.37
17	CMERRY S00A - CASES	$52.37
17	S00A WATER - CASES	$52.37
25	00G F000 - CASES	$101.26
25	CAT F000 - CASES	$101.26
10	SOAP POWDER - CASES	$72.89
10	0ETERGENT - CASES	$72.89
12	HAM - TIMS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORMEO BEEF	$33.75
12	ROAST BEEF	$33.75
1.000	BREA0 - LOAF	$150.00
4.000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	0ETERGEMT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORMEO 8EEF	$33.75
12	ROAST 8EEF	$33.75
1.000	8REA0 - LOAF	$150.00
4.000	ROLLS	$150.00
200	MILK - 0UARTS	$57.42
50	MILK - GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUMCHES	$11.23
10	0ETERGEMT - CASES	$T2.89
12	HAM - TIMS	$36.75
1.000	8REA0 - LOAF	$150.00

OTY	MAME	AMT
4.000	ROLLS	$150.00
200	MILK - 0UARTS	$5T.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	0ETERGEMT - CASES	$72.89
12	HAM - TIMS	$36.75
12	MAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGMA	$33.75
12	CORMEO 8EEF	$33.75
12	ROAST BEEF	$33.75
1.000	BREA0 - LOAF	$150.00
4.000	ROLLS	$150.00
200	MILK - 0UARTS	$57.42
100	MILK - MALF GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - 8UMCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
	TOTAL	$3,893.25

STAMOISH MOTORS
10 WATER STREET
PLYMOUTM. MASS.02296

OTY	MAME	AMT
	PREVIOUS 8ALANCE	$2,356.36
20	AIR CLEANERS - CASES	$200.03
6	GREASE - BARRELS	$165.24
20	TIRES - 650 X 13	$260.38
50	TIRES - 750 X 14	$900.53
50	TIRES - 800 X 14	$1,012.00
100	GASOLINE CAPS	$99.68
	TOTAL	$4,994.22

Sample Problem 2: Console Printer Log and New Master Card Listing

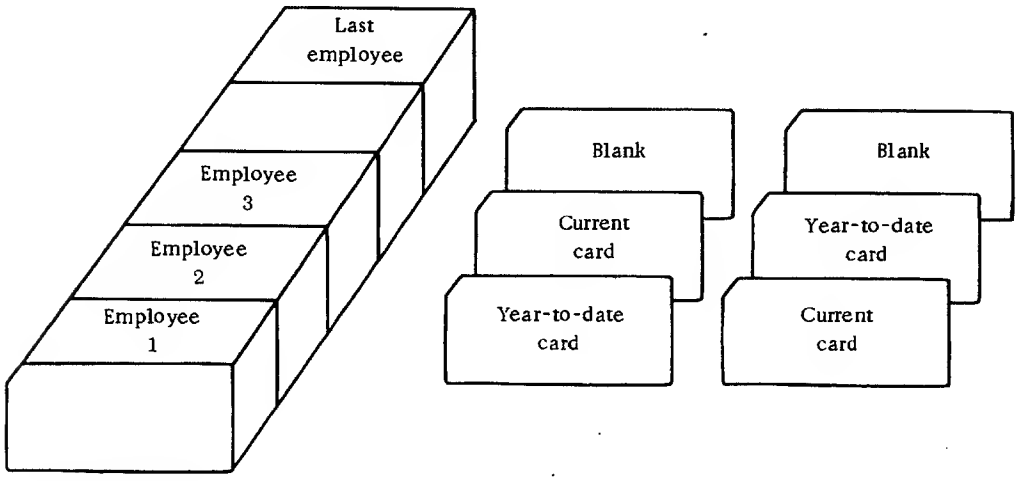
```
ERROR THIS IS A DELIBERATE ERROR                                        J  CSP1404D
ERROR DAVE MARKET                  THIS CARD IS A DELIBERATE MISTAKE    J  CSP14060
DAVES MARKET          1997 WASHINGTUN ST. NEWTOWN, MASS. 02158D0389325 A  CSP14D50
ERRDR STANDISH MOTOR       THIS CARD IS NOT CORRECT   ABCDEFGHIJKLMNOPQRSTUVJ  CSP14850
STANDISH MDTDRS       10 WATER STREET      PLYMDUTH, MASS.022960D499422 A  CSP14790
END DF JOB
```

Sample Problem 2: Data Input Listing

```
END OF JOB                                                                  CSP13960
ERROR                                                                       CSP13970
     , $.  CR                                                               CSP13980
PREVIOUS BALANCE                                                            CSP13990
        QTY          NAME                                                   CSP14000
AMT                                                                         CSP14010
ISTOP                                                                       CSP14020
TOTAL                                                                       CSP14030
THIS IS A DELIBERATE ERROR                                              J   CSP14040
OAVES MARKET        1997 WASHINGTON ST. NEWTOWN, MASS. 02158000111129   A   CSP14050
OAVE MARKET                THIS CARD IS A DELIBERATE MISTAKE            J   CSP14060
DAVES MARKET        SUGAR - BAGS          000021020008                  J   CSP14070
DAVES MARKET        CHICKEN SOUP - CASES000038760011                    J   CSP14080
DAVES MARKET        TOMATO SOUP - CASES 000030110010                    J   CSP14090
OAVES MARKET        SUGAR RETURNED        0000210K0008                  J   CSP14100
OAVES MARKET        COOKIES - CASES       000045210006                  J   CSP14110
DAVES MARKET        GINGER ALE - CASES    000052370017                  J   CSP14120
DAVES MARKET        ROOT BEER - CASES     000052370017                  J   CSP14130
OAVES MARKET        ORANGE ADE - CASES    000052370017                  J   CSP14140
DAVES MARKET        CREME SODA - CASES    000052370017                  J   CSP14150
OAVES MARKET        CHERRY SODA - CASES   000052370017                  J   CSP14160
OAVES MARKET        SODA WATER - CASES    000052370017                  J   CSP14170
DAVES MARKET        DOG FOOD - CASES      000101260025                  J   CSP14180
OAVES MARKET        CAT FOOD - CASES      000101260025                  J   CSP14190
DAVES MARKET        SOAP POWDER - CASES   000072890010                  J   CSP14200
DAVES MARKET        DETERGENT - CASES     000072890010                  J   CSP14210
DAVES MARKET        HAM - TINS            000036750012                  J   CSP14220
DAVES MARKET        HAM - LOAF            000033750012                  J   CSP14230
DAVES MARKET        SALAMI                000033750012                  J   CSP14240
OAVES MARKET        BOLOGNA               000033750012                  J   CSP14250
DAVES MARKET        CORNED BEEF           000033750012                  J   CSP14260
DAVES MARKET        ROAST BEEF            000033750012                  J   CSP14270
OAVES MARKET        BREAD - LOAF          000150001000                  J   CSP14280
DAVES MARKET        ROLLS                 000150004000                  J   CSP14290
DAVES MARKET        MILK - QUARTS         000057420200                  J   CSP14300
DAVES MARKET        MILK - HALF GALS      000057420100                  J   CSP14310
OAVES MARKET        MILK - GALS           000057420050                  J   CSP14320
OAVES MARKET        POTATOES - BAGS       000011230100                  J   CSP14330
DAVES MARKET        TOMATOES - LOOSE      000011230100                  J   CSP14340
DAVES MARKET        CARROTS - BUNCHES     000011230100                  J   CSP14350
DAVES MARKET        DETERGENT - CASES     000072890010                  J   CSP14360
DAVES MARKET        HAM - TINS            000036750012                  J   CSP14370
DAVES MARKET        HAM - LOAF            000033750012                  J   CSP14380
DAVES MARKET        SALAMI                000033750012                  J   CSP14390
OAVES MARKET        BOLOGNA               000033750012                  J   CSP14400
DAVES MARKET        CORNED BEEF           000033750012                  J   CSP14410
DAVES MARKET        ROAST BEEF            000033750012                  J   CSP14420
OAVES MARKET        BREAD - LOAF          000150001000                  J   CSP14430
DAVES MARKET        ROLLS                 000150004000                  J   CSP14440
DAVES MARKET        MILK - QUARTS         000057420200                  J   CSP14450
DAVES MARKET        MILK - GALS           000057420050                  J   CSP14460
DAVES MARKET        MILK - HALF GALS      000057420100                  J   CSP14470
DAVES MARKET        POTATOES - BAGS       000011230100                  J   CSP14480
OAVES MARKET        TOMATOES - LOOSE      000011230100                  J   CSP14490

OAVES MARKET        CARROTS - BUNCHES     000011230100                  J   CSP14500
DAVES MARKET        DETERGENT - CASES     000072890010                  J   CSP14510
DAVES MARKET        HAM - TINS            000036750012                  J   CSP14520
OAVES MARKET        BREAD - LOAF          000150001000                  J   CSP14530
OAVES MARKET        ROLLS                 000150004000                  J   CSP14540
DAVES MARKET        MILK - QUARTS         000057420200                  J   CSP14550
DAVES MARKET        MILK - HALF GALS      000057420100                  J   CSP14560
DAVES MARKET        MILK - GALS           000057420050                  J   CSP14570
DAVES MARKET        POTATOES - BAGS       000011230100                  J   CSP14580
OAVES MARKET        TOMATOES - LOOSE      000011230100                  J   CSP14590
DAVES MARKET        CARROTS - BUNCHES     000011230100                  J   CSP14600
DAVES MARKET        DETERGENT - CASES     000072890010                  J   CSP14610
DAVES MARKET        HAM - TINS            000036750012                  J   CSP14620
OAVES MARKET        HAM - LOAF            000033750012                  J   CSP14630
DAVES MARKET        SALAMI                000033750012                  J   CSP14640
OAVES MARKET        BOLOGNA               000033750012                  J   CSP14650
DAVES MARKET        CORNED BEEF           000033750012                  J   CSP14660
DAVES MARKET        ROAST BEEF            000033750012                  J   CSP14670
DAVES MARKET        BREAD - LOAF          000150001000                  J   CSP14680
DAVES MARKET        ROLLS                 000150004000                  J   CSP14690
DAVES MARKET        MILK - QUARTS         000057420200                  J   CSP14700
OAVES MARKET        MILK - HALF GALS      000057420100                  J   CSP14710
OAVES MARKET        MILK - HALF GALS      000057420100                  J   CSP14720
OAVES MARKET        POTATOES - BAGS       000011230100                  J   CSP14730
DAVES MARKET        TOMATOES - LOOSE      000011230100                  J   CSP14740
DAVES MARKET        CARROTS - BUNCHES     000011230100                  J   CSP14750
OAVES MARKET        DETERGENT - CASES     000072890010                  J   CSP14760
DAVES MARKET        HAM - TINS            000036750012                  J   CSP14770
                                                                          A   CSP14780
STANDISH MOTORS     10 WATER STREET       PLYMOUTH, MASS.0229600235636  A   CSP14790
STANDISH MOTORS     AIR CLEANERS - CASES000200030020                    J   CSP14800
STANDISH MOTORS     GREASE - BARRELS      000165240006                  J   CSP14810
STANDISH MOTORS     TIRES - 650 X 13      000260380020                  J   CSP14820
STANDISH MOTORS     TIRES - 750 X 14      000900530050                  J   CSP14830
STANDISH MOTORS     TIRES - 800 X 14      001012000050                  J   CSP14840
STANDISH MOTOR      THIS CARD IS NOT CORRECT  ABCDEFGHIJKLMNOPQRSTUVWJ  CSP14850
STANDISH MOTORS     GASOLINE CAPS         000099680100                  J   CSP14860
                                                                          A   CSP14870
ISTOP                                                                       CSP14880
```

PROBLEM 3

The purpose of this program is to print a payroll register and punch a new year-to-date card for each employee. The input deck is as follows:



The year-to-date and current cards are read and processed. The payroll register is printed as in the example, and a new year-to-date card image is printed on the console printer. Then the next employee is processed.

As is shown, the order of the year-to-date card and current card is not known before the cards are read.

If the user's system has an 1132 Printer, switch 0 on the console must be in the up position, and all other switches in the down position. If the user's system does not have an 1132 Printer, all switches on the console must be in the down position.

Sample Problem 3: Detailed Description

1. Determine the output unit from the data switches.

   0=console printer, 1=1132 Printer

2. Read the edit mask.

3. Read a card.

4. Is the card read in (3) blank?

   Yes — 18 No — 5

5. Is the card read in (3) a year-to-date card?

   Yes — 11 No — 6

6. Is the card read in (3) a current card?

   Yes — 8 No — 7

7. Stop.

8. Move the employee number to storage (JEMP).

9. Extract the number of hours worked (HRS).

10. Go to (3).

11. Move the department number to storage (IDEP).

12. Move the employee number to storage (IEMP).

13. Move the employee name to storage (INM).

14. Move the Social Security number to storage (ISS).

15. Move the pay rate to storage (IRT).

16. Move the year-to-date gross to storage (IYTD).

17. Go to (3).

18. Are IEMP and JEMP the same?

    Yes — 19 No — 24

19. Current amount (CURR) is set equal to HRS times pay rate.

20. New year-to-date is set equal to CURR +IYTD.

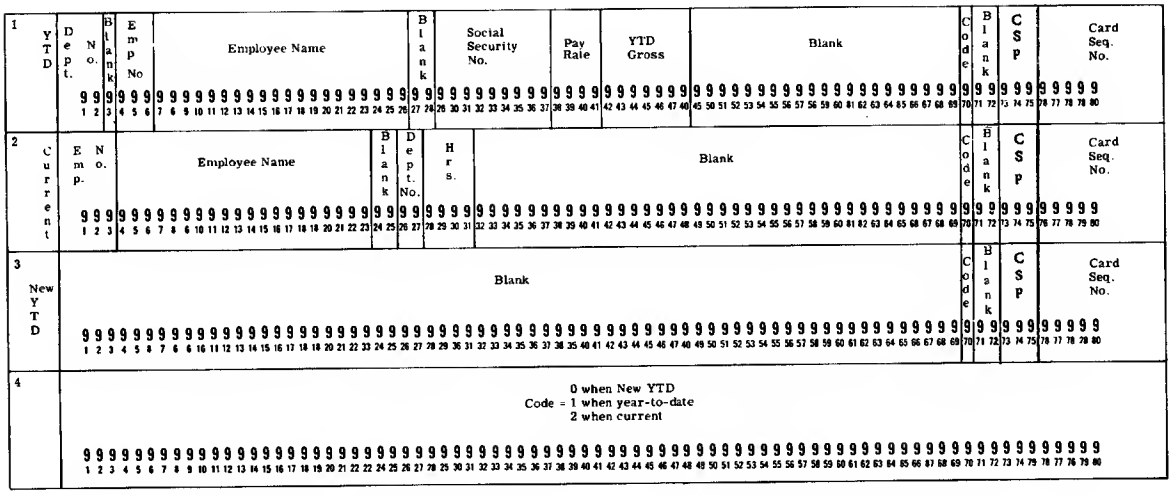
21. Print a new year-to-date card image on the console printer.

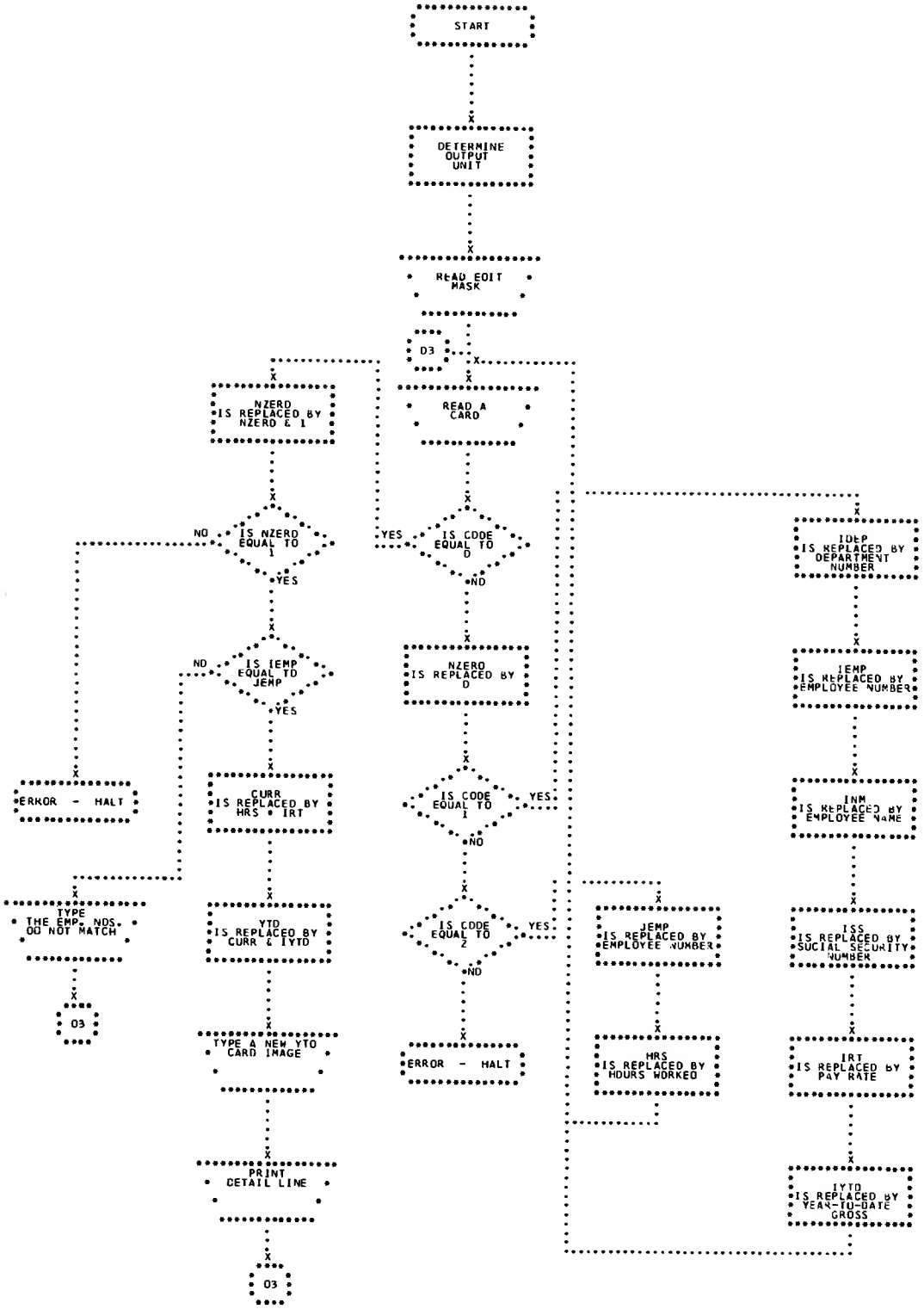
22. Print the payroll register line as in the example.

23. Go to (3).

24. Halt. If start is pushed, go to (3).

Card Formats




START
DETERMINE OUTPUT UNIT
READ EDIT MASK
D3
READ A CARD
IS CODE EQUAL TO D
YES
NO
NZERO IS REPLACED BY D
IS CODE EQUAL TO I
IS CODE EQUAL TO 2
ERROR - HALT
NZERD IS REPLACED BY NZERD & 1
IS NZERD EQUAL TO 1
IS IEMP EQUAL TO JEMP
ERROR - HALT
TYPE THE EMP. NOS. DO NOT MATCH
03
CURR IS REPLACED BY HRS * IRT
YTD IS REPLACED BY CURR & IYTD
TYPE A NEW YTD CARD IMAGE
PRINT DETAIL LINE
03
IDEP IS REPLACED BY DEPARTMENT NUMBER
IEMP IS REPLACED BY EMPLOYEE NUMBER
INM IS REPLACED BY EMPLOYEE NAME
ISS IS REPLACED BY SOCIAL SECURITY NUMBER
IRT IS REPLACED BY PAY RATE
IYTD IS REPLACED BY YEAR-TO-DATE GROSS
JEMP IS REPLACED BY EMPLOYEE NUMBER
HRS IS REPLACED BY HOURS WORKED

Sample Problem 3: Source Program

```
// FOR                                                                  CSP14890


                                                                        PAGE 01

**  SAMPLE PROBLEM 3                                                    C5P14900
*  NAME 5P3                                                             C5P14910
*  IOCS(CARD,TYPEWRITER,1132 PRINTER)                                   C5P14920
*  ONE WORD INTEGERS                                                    C5P14930
*  EXTENDED PRECISION                                                   C5P14940
*  LIST ALL                                                             C5P14950


 SAMPLE PROBLEM 3                                                       PAGE 02

      DIMENSION MASK(12),IN(69),IDEP(2),IEMP(3),INM(20),ISS(9),IRT(4),  C5P14960
     1 IYTD(7),JEMP(3),NYTO(7),ICUR(6),KCURR(12),KOYTD(12),KNYTO(12)    CSP14970
    1 FORMAT (69A1,I1)                                                  C5P14980
    2 FORMAT (12A1)                                                     CSP14990
20    FORMAT (1H ,2A1,1X,23A1,2X,20A1,21X,1H1,3X,7HC5P                  C5P15000
30    FORMAT (1H ,2A1,2X,3A1,2X,20A1,5X,3(12A1,2X))                     CSP15010
      CALL OATSW(0,I)                                                   C5P15020
      NREAD=2                                                           C5P15030
      NWRIT=2*(1/I)+1                                                   C5P15040
      READ (NREAD,2) MASK                                               C5P15050
   15 READ (NREAD,1) IN,ICD                                             C5P15060
      IF (ICD) 6,10,6                                                   C5P15070
6     NZERO=0                                                           CSP15080
      GO TO (7,8), ICD                                                  CSP15090
C     THIS IS THE YEAR TO DATE PROCESSING                               C5P15100
    7 CALL MOVE (IN,1,IDEP,1)                                           C5P15110
      CALL MOVE (IN,4,6,IEMP,1)                                         C5P15120
      CALL MOVE (IN,7,26,INM,1)                                         CSP15130
      CALL MOVE (IN,29,37,ISS,1)                                        CSP15140
      CALL MOVE (IN,38,41,IRT,1)                                        CSP15150
      CALL MOVE (IN,42,48,IYTD,1)                                       C5P15160
      GO TO 15                                                          CSP15170
C     THIS IS CURRENT PERIOD PROCESSING                                 C5P15180
    8 CALL MOVE (IN,1,3,JEMP,1)                                         C5P15190
      HRS=GET (IN,28,30,100,0)                                          CSP15200
      GO TO 15                                                          CSP15210
10    NZERO = NZERO + 1                                                 C5P15220
      IF (NZERO - 1) 100,100,101                                        CSP15230
101   STOP                                                              CSP15240
100   IF (NCOMP(IEMP,1,3,JEMP,1)) 99,11,99                             CSP15250
   11 CURR=(HRS*GET(IRT,1,4,10,0)+500,0)/1000,0                         C5P15260
      YTO=CURR+GET (IYTD,1,7,10,0)                                      CSP15270
      CALL PUT (NYTD,1,7,YTD,5,0,1)                                     CSP15280
      WRITE (1,20( IDEP,IEMP,INM,ISS,IRT,NYTD                           C5P15290
      CALL PUT (ICUR,1,6,CURR,5,0,1)                                    CSP15300
      CALL MOVE (MASK,1,12,KCURR,1)                                     CSP15310
      CALL MOVE (MASK,1,12,KOYTO,1)                                     CSP15320
      CALL MOVE (MASK,1,12,KNYTD,1I                                     CSP15330
      CALL EDIT (ICUR,1,6,KCURR,1,12)                                   CSP15340
      CALL EDIT (IYTD,1,7,KOYTD,1,12)                                   C5P15350
      CALL EDIT (NYTO,I,7,KNYTD,1,12I                                   CSP15360
      WRITE (NWRIT,30) IDEP,IEMP,INM,KOYTO,KCURR,KNYTD                  CSP15370
      GO TO 15                                                          CSP15380
C     THIS IS AN ERROR.  THE EMP NOS DO NOT MATCH.                      CSP15390
99    WRITE (1,40)                                                      CSP15400
40    FORMAT (' THE EMP NOS OO NOT MATCH,')                             CSP15410
      GO TO 15                                                          CSP15420
      END                                                               C5P15430


 SAMPLE PROBLEM 3                                                       PAGE 03

VARIABLE ALLOCATIONS
 HRS  =0000  CURR =0003  YTD  =0006  MASK =0014  IN   =0059  IDEP =005B  IEMP =005E  INM  =0072  ISS  =007B  IRT  =007F
 IYTD =0086  JEMP =0089  NYTD =0090  ICUR =0096  KCURR=00A2  KOYTD=00AE  KNYTD=00BA  I    =00BB  NREAD=00BC  NWRIT=00BD
 ICO  =00BE  NZERO=00BF

STATEMENT ALLOCATIONS
 1    =00E1  2    =00E5  20   =00E8  30   =00FC  40   =010D  15   =0149  6    =0155  7    =015F  8    =018B  10   =019C
 101  =01A8  100  =01AA  11   =01B3  99   =0236

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CALLED SUBPROGRAMS
 OATSW   MOVE    GET     NCOMP   PUT     EDIT    EADD    EMPY    EDIV    ELD     ESTO    WRTYZ   SRED    SWRT    SCOMP
 SFIO    SIOAI   SIOI    STOP    CARDZ   PRNTZ

REAL CONSTANTS
   .100000000E 03=00C0     .100000000E 02=00C3     .500000000E 03=00C6     .100000000E 04=00C9     .500000000E 01=00CC

INTEGER CONSTANTS
     0=00CF       2=00D0       1=00D1       4=0002       6=0003       7=0004      26=00D5      29=0006      37=00D7      38=00D8
    41=00D9      42=00DA      48=00DB       3=00DC      28=00DD      30=000E      12=000F       0=00E0

CORE REQUIREMENTS FOR SP3
 COMMON       0  VARIABLES    192  PROGRAM    380

END OF COMPILATION
```

Sample Problem 3: Payroll Register Output

```
// XEO                                                              CSP15440

01  101  KCINRAM.  S          $7.453.06      $198.91      $7.651.97
52  201  OMINOREG.     M      $3.524.37      $143.82      $3.668.19
76  676  NEDAB.    R          $10.060.60     $297.27      $10.357.87
76  689  NEOUOL.   R          $10.060.60     $297.27      $10.357.87
01  253  ECAM.     O          $9.555.62      $279.65      $9.835.27
```

Sample Problem 3: Console Printer Error Log and New Year-to-Date Card Image

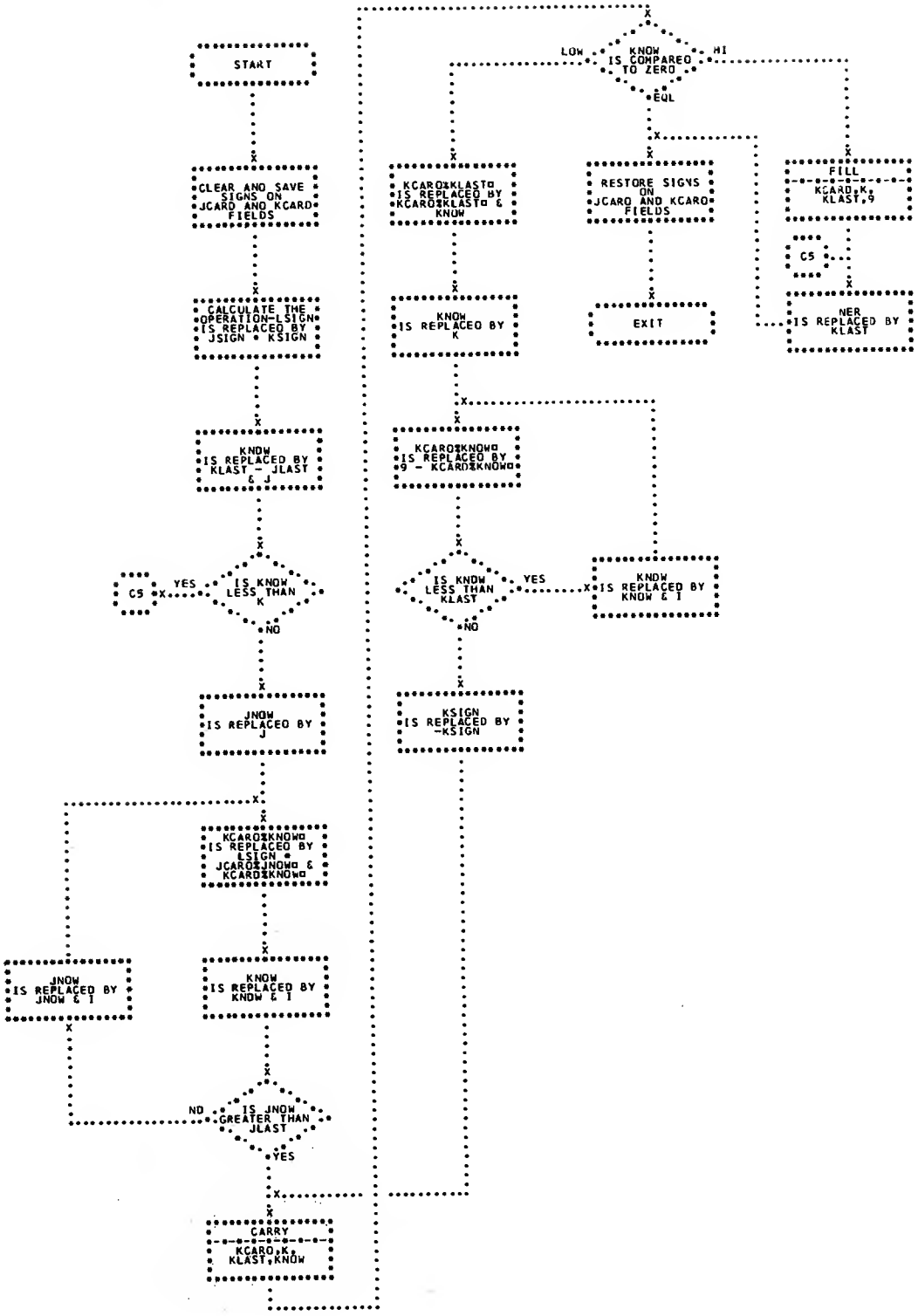
```
01 101KCINRAH,  S          79856643205420765197                1   CSP

52 2010HINOREG,    M       01332567804230366819                1   CSP

76 676NEOA8,    R          01423306008101035787                1   CSP

76 689NEOUOL,   R          79860379408101035787                1   CSP

THE EMP NOS OO NOT MATCH.

01 253ECAM,     O          95462305707620983527                1   CSP
```

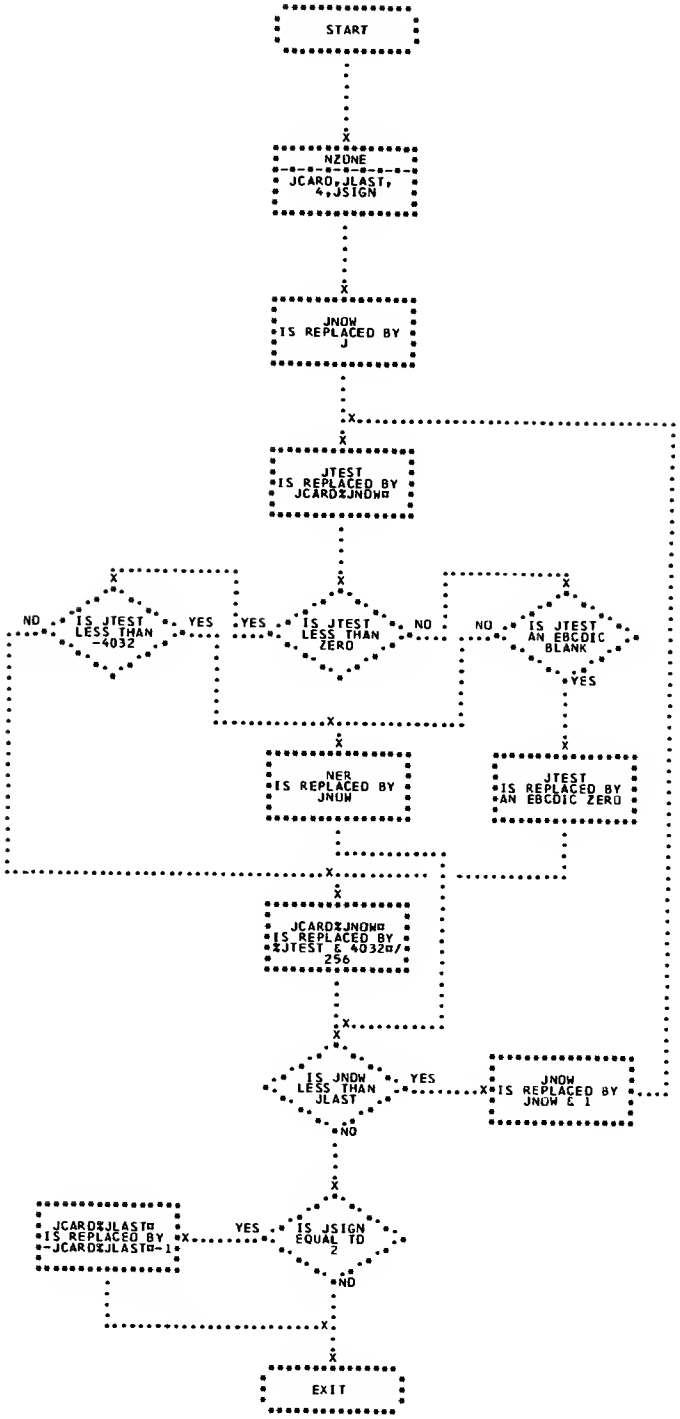
Sample Problem 3: Data Input Listing

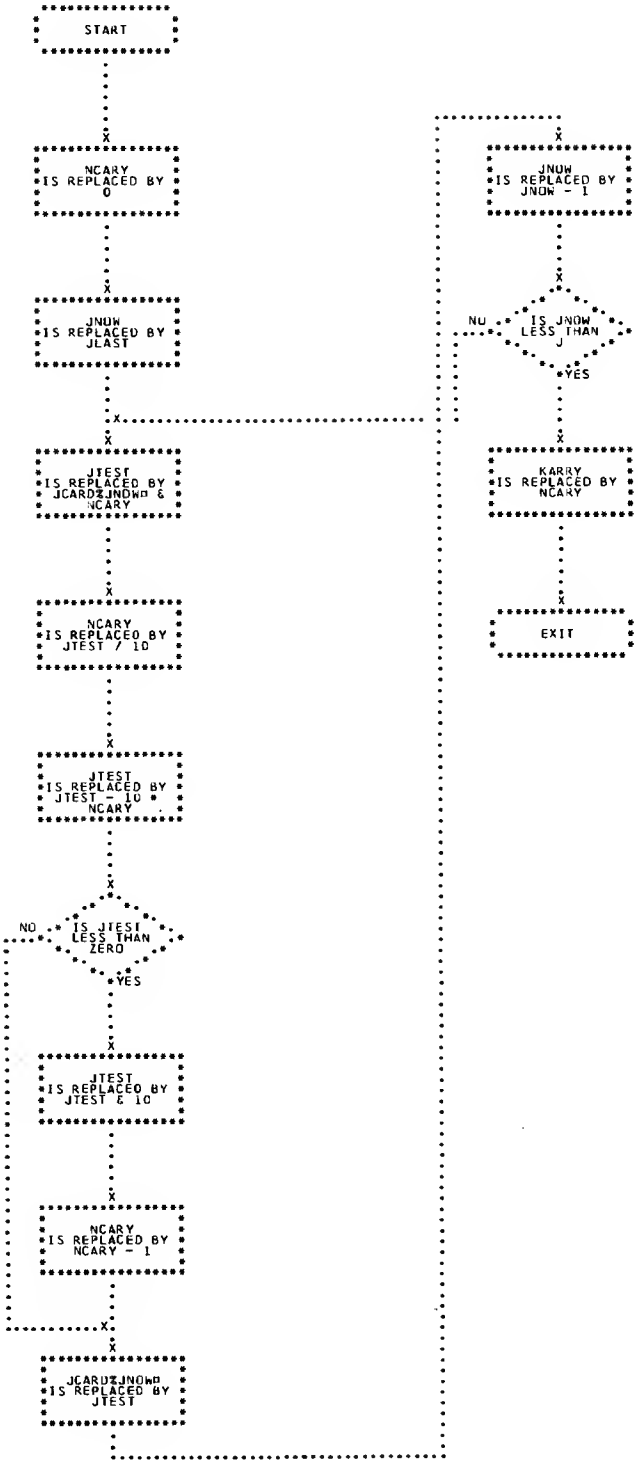
```
    ,  S,  CR                                                    CSP15450
01 101KCINRAH,  S          79856643205420745306                1 CSP15460
101KCINRAH,  S          01367                                  2 CSP15470
                                                               0 CSP15480
2010MINOREG,     M      52340                                  2 CSP15490
52 2010MINOREG,     M       01332567804230352437               1 CSP15500
                                                               0 CSP15510
76 676NEOAB,    R          01423306008101006060                1 CSP15520
676NEOAB,    R          76367                                  2 CSP15530
                                                               0 CSP15540
689NEOUOL,    R         76367                                  2 CSP15550
76 689NEOUOL,    R          79860379408101006060               1 CSP15560
                                                               0 CSP15570
99 999NIVOEN,    A          99999999901160511122                1 CSP15580
099NIVDEN,    A         994009                                 2 CSP15590
                                                               0 CSP15600
01 253ECAM,     O          95462305707620955562                1 CSP15610
253ECAM,     D          01367                                  2 CSP15620
                                                               0 CSP15630
                                                                 CSP15640
```

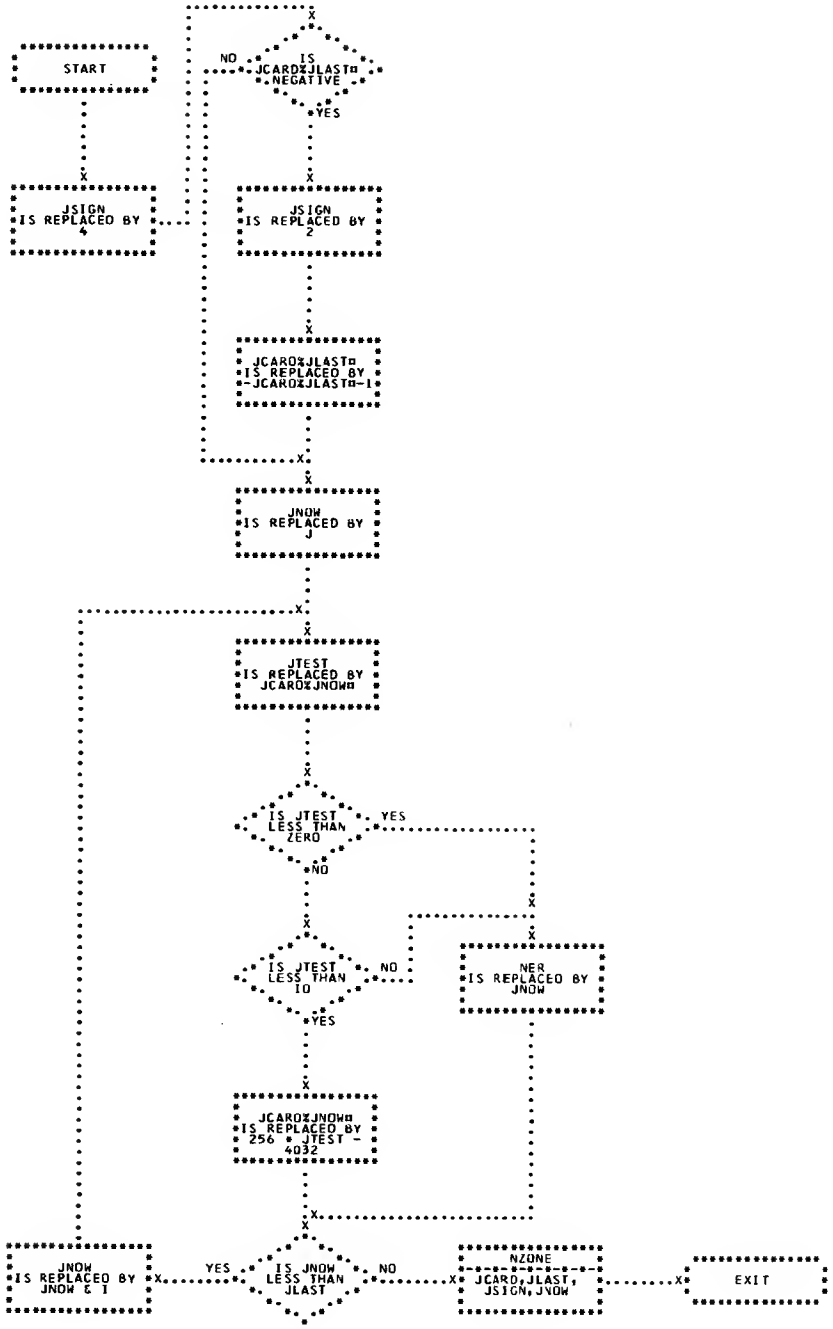
FLOWCHARTS

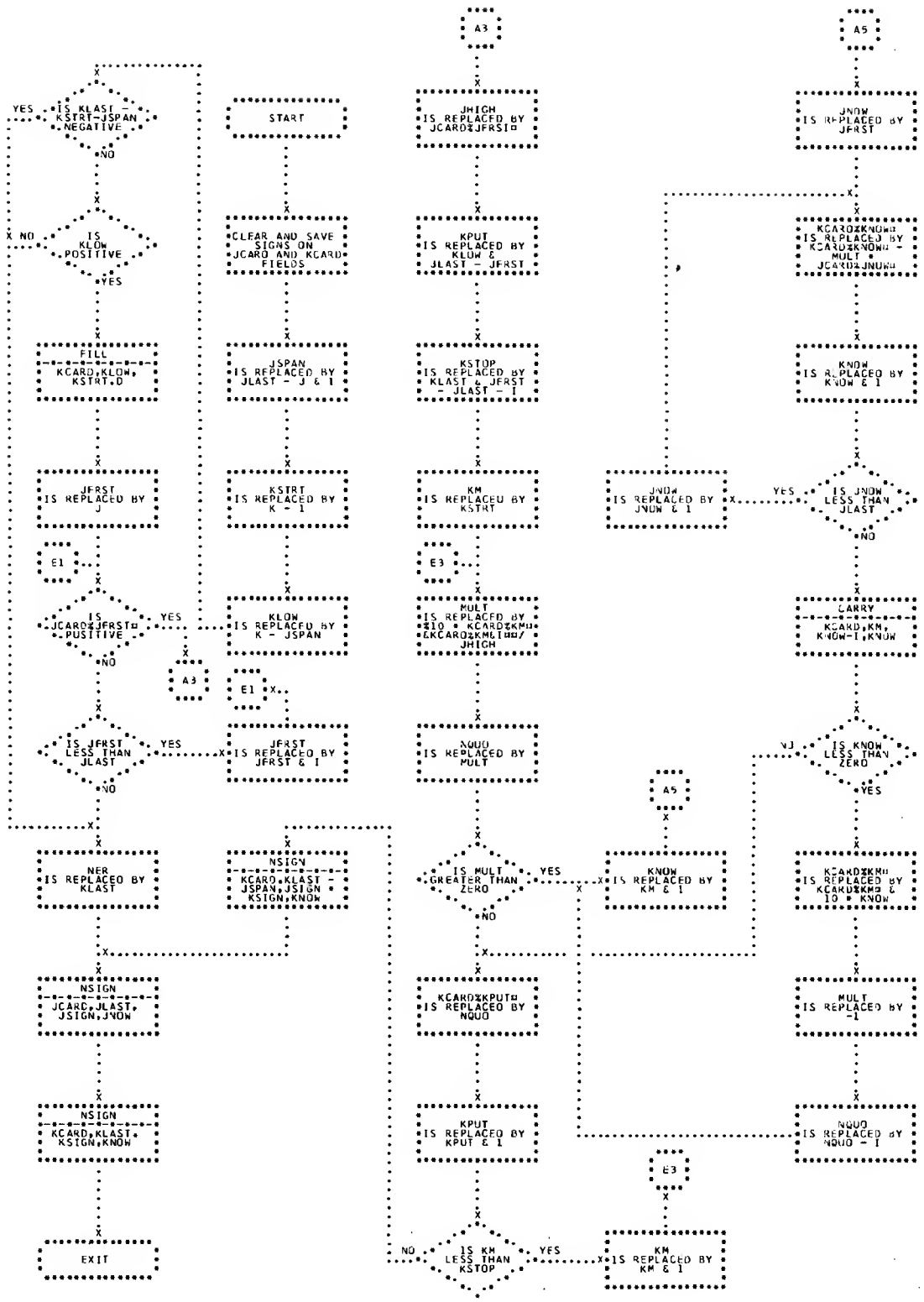
CHART AD 1130 COMMERCIAL ADD SUBROUTINE

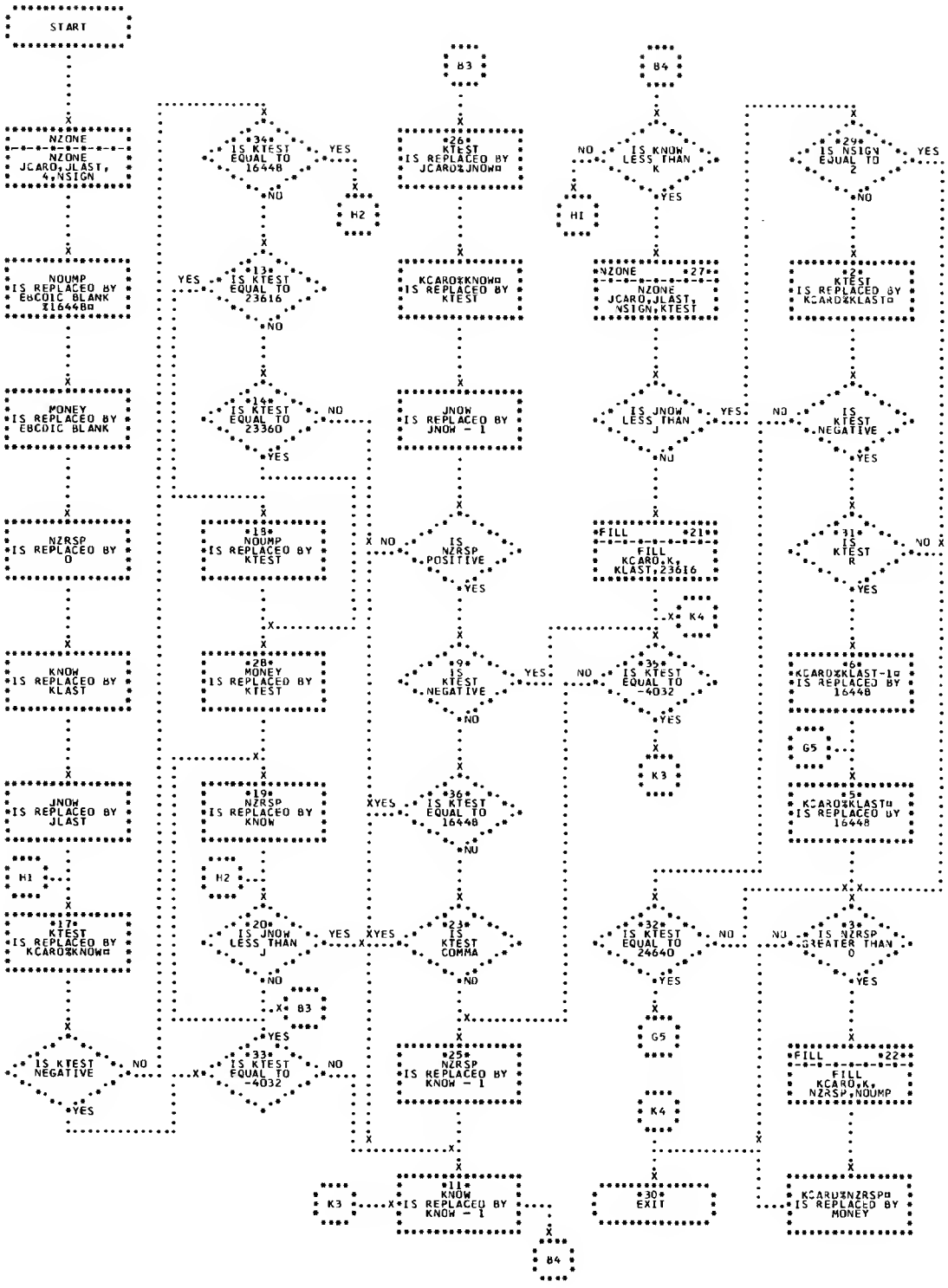


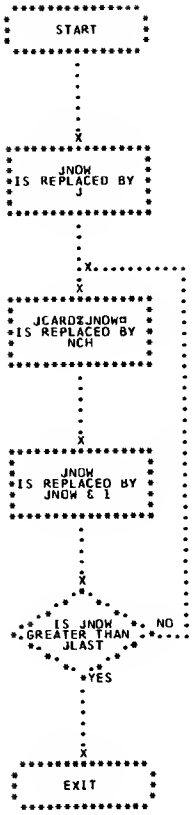

START
NZONE
JCARD,JLAST,
4,JSIGN
JNOW
IS REPLACED BY
J
JTEST
IS REPLACED BY
JCARD¤JNOW¤
IS JTEST
LESS THAN
-4032
NO
YES
IS JTEST
LESS THAN
ZERO
YES
NO
IS JTEST
AN EBCDIC
BLANK
NO
YES
NER
IS REPLACED BY
JNOW
JTEST
IS REPLACED BY
AN EBCDIC ZERO
JCARD¤JNOW¤
IS REPLACED BY
¤JTEST ¢ 4032¤/¤
256
IS JNOW
LESS THAN
JLAST
YES
NO
JNOW
IS REPLACED BY
JNOW ¢ 1
JCARD¤JLAST¤
IS REPLACED BY
-JCARD¤JLAST¤-1
IS JSIGN
EQUAL TO
2
YES
NO
EXIT

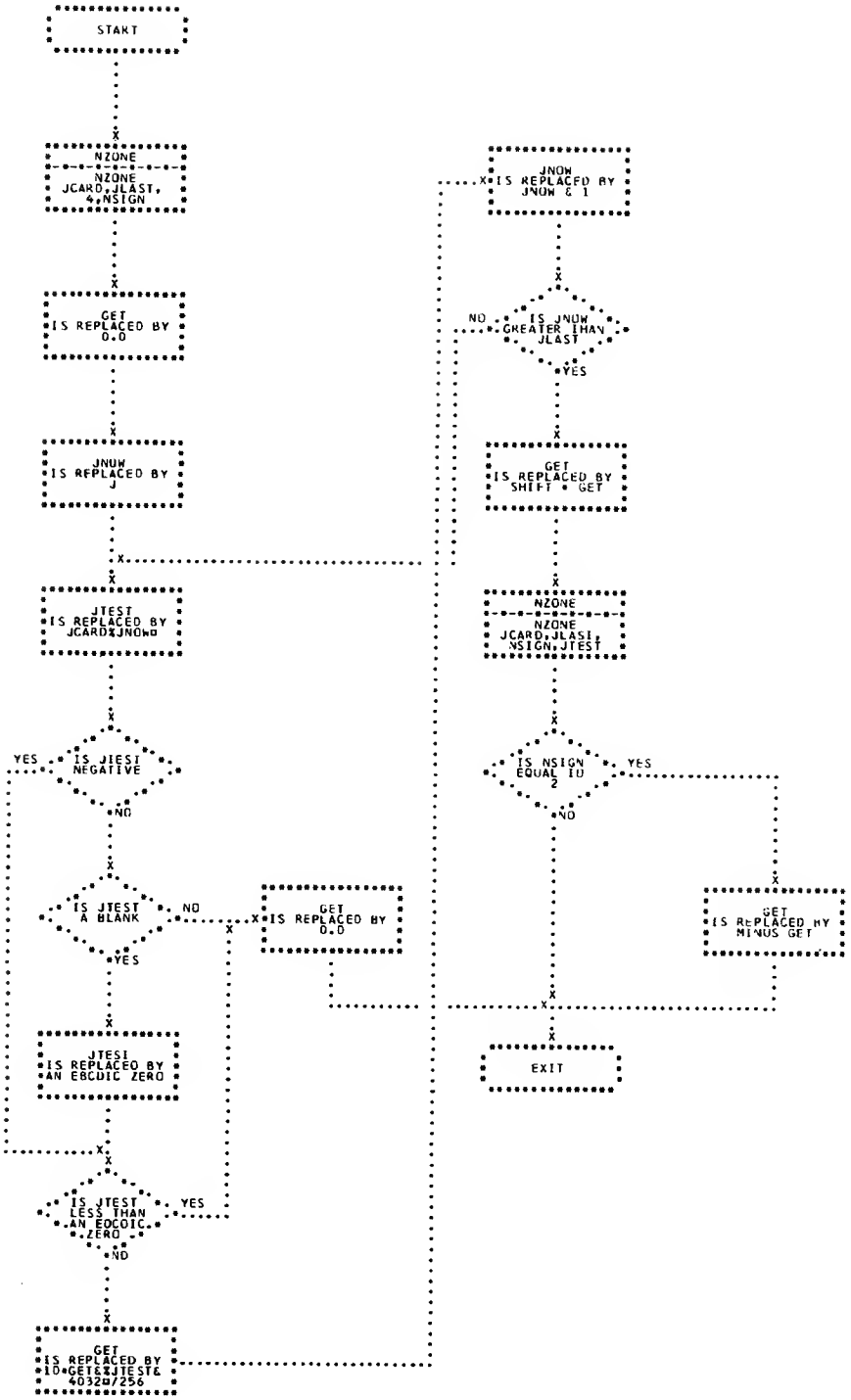

START
NCARY IS REPLACED BY 0
JNOW IS REPLACED BY JLAST
JTEST IS REPLACED BY JCARD¤JNOW¤ & NCARY
NCARY IS REPLACED BY JTEST / 10
JTEST IS REPLACED BY JTEST - 10 * NCARY
IS JTEST LESS THAN ZERO
NO
YES
JTEST IS REPLACED BY JTEST & 10
NCARY IS REPLACED BY NCARY - 1
JCARD¤JNOW¤ IS REPLACED BY JTEST
JNOW IS REPLACED BY JNOW - 1
IS JNOW LESS THAN J
NO
YES
KARRY IS REPLACED BY NCARY
EXIT

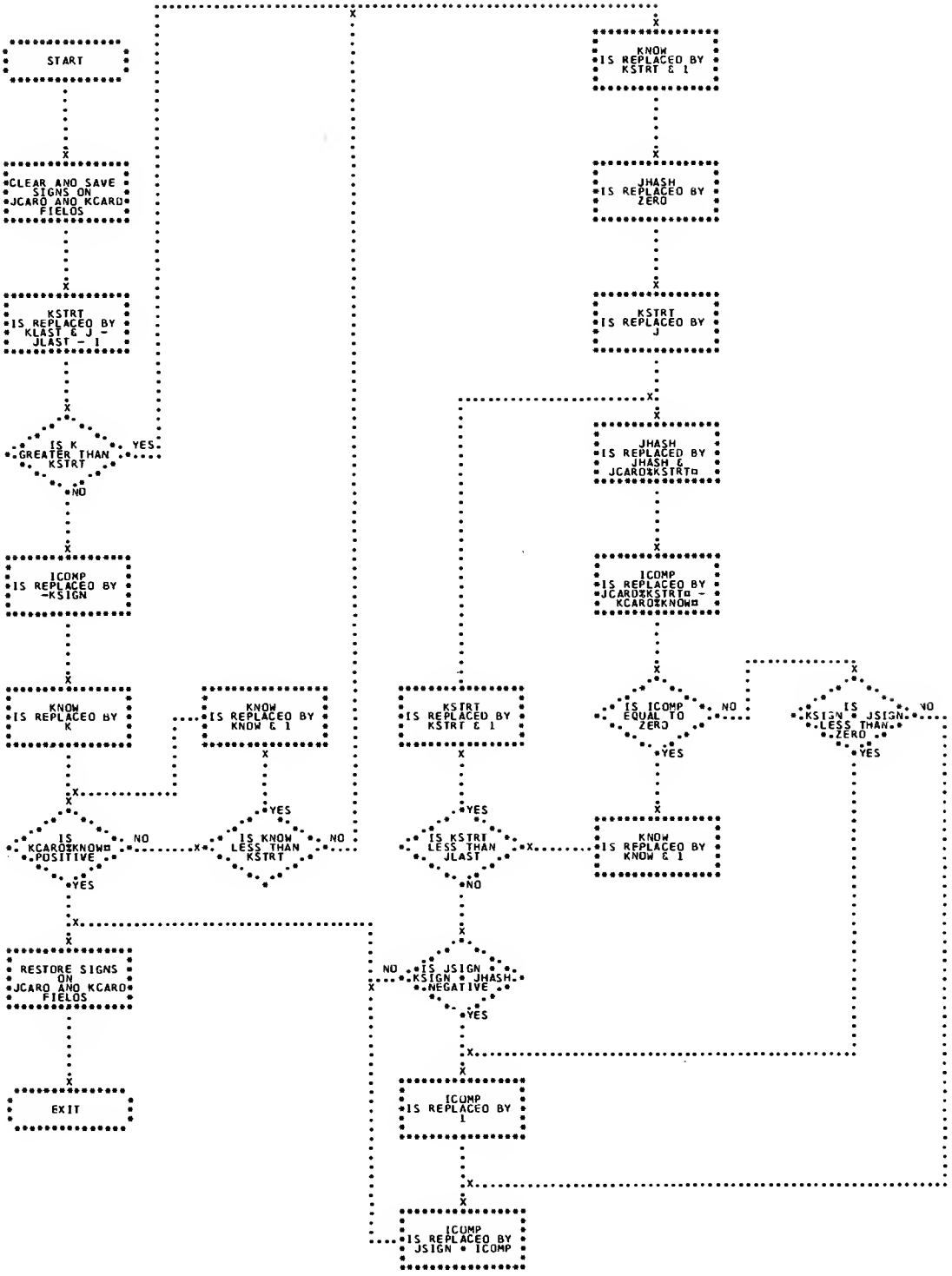

START
IS JCARD¤JLAST¤ NEGATIVE
NO
YES
JSIGN IS REPLACED BY 4
JSIGN IS REPLACED BY 2
JCARD¤JLAST¤ IS REPLACED BY -JCARD¤JLAST¤-1
JNOW IS REPLACED BY J
JTEST IS REPLACED BY JCARD¤JNOW¤
IS JTEST LESS THAN ZERO
YES
NO
IS JTEST LESS THAN 10
NO
YES
NER IS REPLACED BY JNOW
JCARD¤JNOW¤ IS REPLACED BY 256 * JTEST - 4032
JNOW IS REPLACED BY JNOW & 1
IS JNOW LESS THAN JLAST
YES
NO
NZONE JCARD,JLAST, JSIGN,JNOW
EXIT

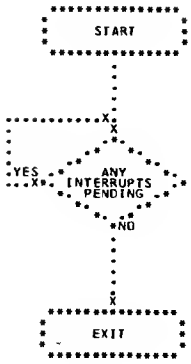

START
CLEAR AND SAVE SIGNS ON JCARD AND KCARD FIELDS
IS KLAST - KSTRT-JSPAN NEGATIVE
YES
NO
IS KLOW POSITIVE
NO
YES
FILL KCARD,KLOW, KSTRT,D
JFRST IS REPLACED BY J
E1
IS JCARD&JFRST¤ POSITIVE
YES
NO
IS JFRST LESS THAN JLAST
YES
NO
JSPAN IS REPLACED BY JLAST - J & 1
KSTRT IS REPLACED BY K - 1
KLOW IS REPLACED BY K - JSPAN
A3
E1
JFRST IS REPLACED BY JFRST & I
NER IS REPLACED BY KLAST
NSIGN KCARD,KLAST - JSPAN,JSIGN KSIGN,KNOW
NSIGN JCARD,JLAST, JSIGN,JNOW
NSIGN KCARD,KLAST, KSIGN,KNOW
EXIT
A3
JHIGH IS REPLACED BY JCARD&JFRS1¤
KPUT IS REPLACED BY KLOW & JLAST - JFRST
KSTOP IS REPLACED BY KLAST & JFRST - JLAST - I
KM IS REPLACED BY KSTRT
E3
MULT IS REPLACED BY &IO * KCARD&KM1¤ &KCARD&KMLT¤¤/ JHIGH
NQUO IS REPLACED BY MULT
IS MULT GREATER THAN ZERO
YES
NO
KCARD&KPUT¤ IS REPLACED BY NQUO
KPUT IS REPLACED BY KPUT & 1
IS KM LESS THAN KSTOP
NO
YES
A5
KNOW IS REPLACED BY KM & 1
A5
E3
KM IS REPLACED BY KM & 1
A5
JNOW IS REPLACED BY JFRST
KCARD&KNOW¤ IS REPLACED BY KCARD&KNOW - MULT * JCARD&JNOW¤
KNOW IS REPLACED BY KNOW & 1
IS JNOW LESS THAN JLAST
YES
NO
JNOW IS REPLACED BY JNOW & 1
CARRY KCARD,KM, KNOW-1,KNOW
IS KNOW LESS THAN ZERO
YES
NO
KCARD&KM¤ IS REPLACED BY KCARD&KM2 & 10 * KNOW
MULT IS REPLACED BY -1
NQUO IS REPLACED BY NQUO - I

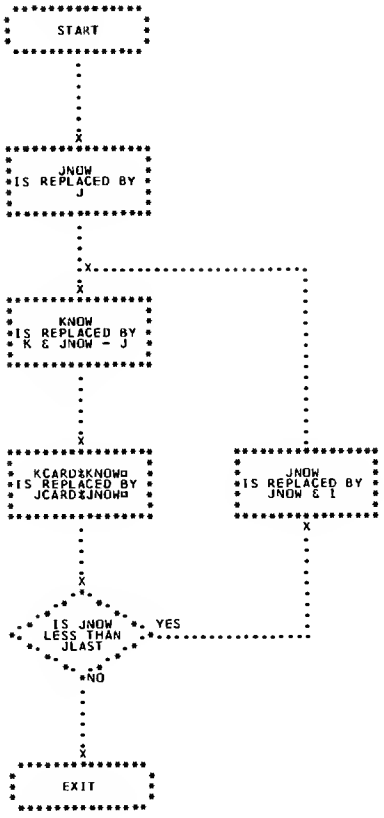

START
NZONE
NZONE JCARD,JLAST, 4,NSIGN
NOUMP IS REPLACED BY EBCDIC BLANK %16448¤
MONEY IS REPLACED BY EBCDIC BLANK
NZRSP IS REPLACED BY 0
KNOW IS REPLACED BY KLAST
JNOW IS REPLACED BY JLAST
H1
17 KTEST IS REPLACED BY KCARD%KNOW¤
IS KTEST NEGATIVE
NO
YES
34 IS KTEST EQUAL TO 16448
YES
NO
H2
13 IS KTEST EQUAL TO 23616
YES
NO
14 IS KTEST EQUAL TO 23360
NO
YES
18 NOUMP IS REPLACED BY KTEST
28 MONEY IS REPLACED BY KTEST
19 NZRSP IS REPLACED BY KNOW
H2
20 IS JNOW LESS THAN J
YES
NO
B3
YES
33 IS KTEST EQUAL TO -4032
NO
YES
B3
26 KTEST IS REPLACED BY JCARD%JNOW¤
KCARD%KNOW¤ IS REPLACED BY KTEST
JNOW IS REPLACED BY JNOW - 1
IS NZRSP POSITIVE
NO
YES
19 IS KTEST NEGATIVE
YES
NO
36 IS KTEST EQUAL TO 16448
YES
NO
23 IS KTEST COMMA
YES
NO
25 NZRSP IS REPLACED BY KNOW - 1
K3
11 KNOW IS REPLACED BY KNOW - 1
B4
B4
IS KNOW LESS THAN K
NO
YES
HI
*NZONE *27* NZONE JCARD,JLAST, NSIGN,KTEST
IS JNOW LESS THAN J
YES
NO
*FILL *21* FILL KCARD,K, KLAST,23616
K4
35 IS KTEST EQUAL TO -4032
NO
YES
K3
32 IS KTEST EQUAL TO 24640
NO
YES
G5
K4
30 EXIT
29 IS NSIGN EQUAL TO 2
YES
NO
2 KTEST IS REPLACED BY KCARD%KLAST¤
IS NEGATIVE
NO
YES
31 IS KTEST R
NO
YES
6 KCARD%KLAST-1¤ IS REPLACED BY 16448
G5
5 KCARD%KLAST¤ IS REPLACED BY 16448
3 IS NZRSP GREATER THAN 0
NO
YES
*FILL *22* FILL KCARD,K, NZRSP,NOUMP
KCARD%NZRSP¤ IS REPLACED BY MONEY

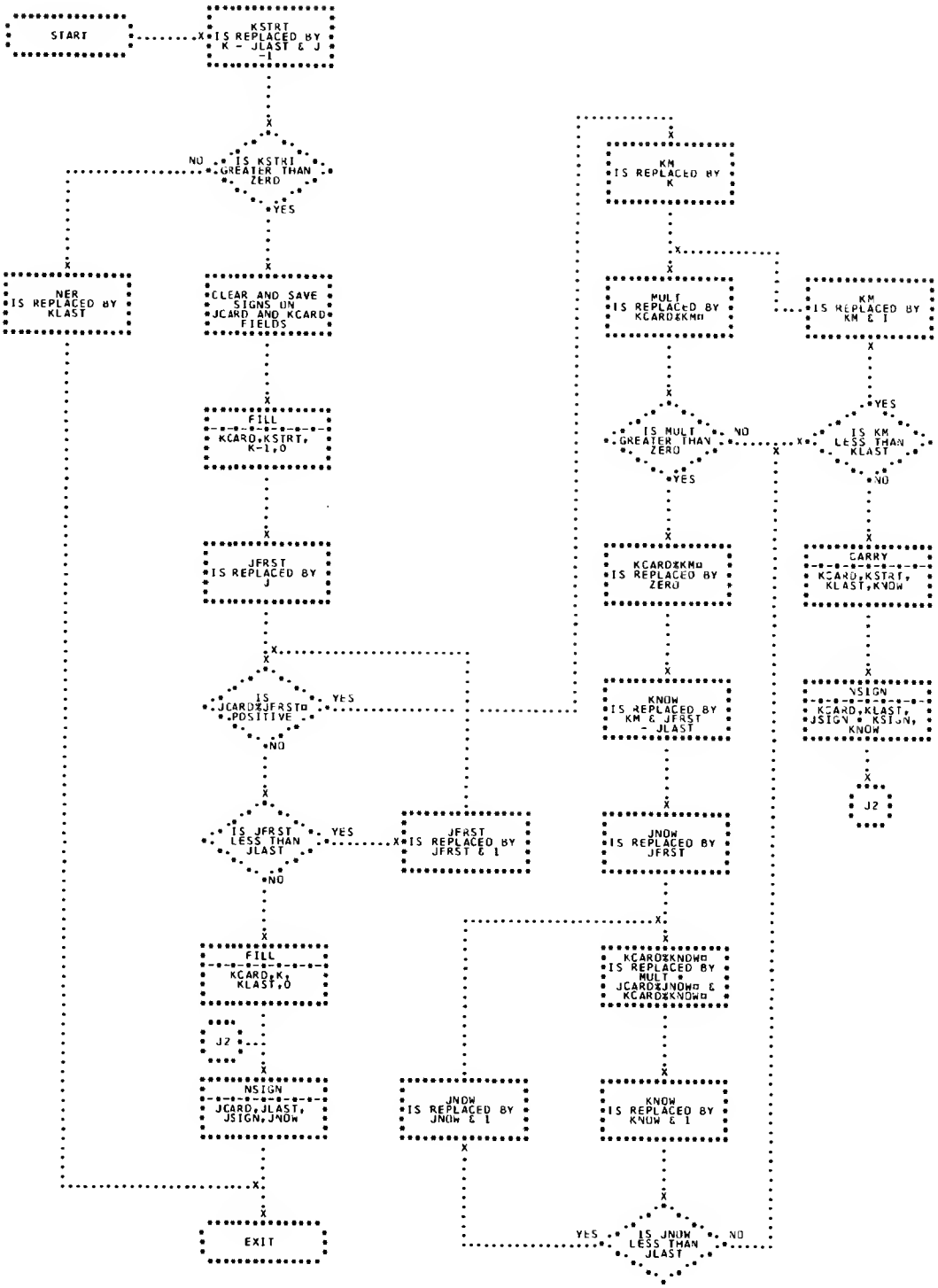

START
JNOW
IS REPLACED BY
J
JCARD≠JNOW≠
IS REPLACED BY
NCH
JNOW
IS REPLACED BY
JNOW & 1
IS JNOW
GREATER THAN
JLAST
NO
YES
EXIT

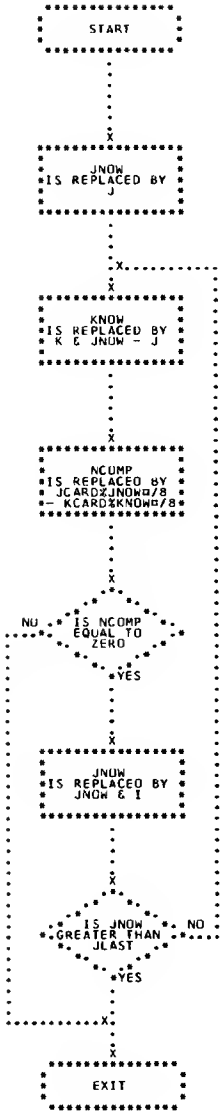

START
NZONE
NZONE JCARD,JLAST, 4,NSIGN
GET IS REPLACED BY 0.0
JNOW IS REPLACED BY J
JTEST IS REPLACED BY JCARD&JNOW
IS JTEST NEGATIVE
YES
NO
IS JTEST A BLANK
NO
YES
GET IS REPLACED BY 0.0
JTEST IS REPLACED BY AN EBCDIC ZERO
IS JTEST LESS THAN AN EBCDIC ZERO
YES
NO
GET IS REPLACED BY ID*GET&JTEST& 4032@7256
JNOW IS REPLACED BY JNOW & 1
IS JNOW GREATER THAN JLAST
NO
YES
GET IS REPLACED BY SHIFT * GET
NZONE
NZONE JCARD,JLAST, NSIGN,JTEST
IS NSIGN EQUAL TO 2
YES
NO
GET IS REPLACED BY MINUS GET
EXIT

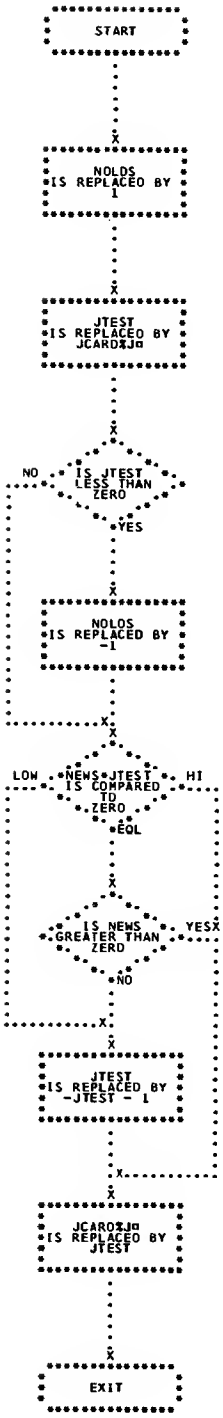

START
CLEAR AND SAVE SIGNS ON JCARD AND KCARD FIELDS
KSTRT IS REPLACED BY KLAST & J - JLAST - 1
IS K GREATER THAN KSTRT
YES
NO
ICOMP IS REPLACED BY -KSIGN
KNOW IS REPLACED BY K
IS KCARD&KNOW¤ POSITIVE
NO
YES
KNOW IS REPLACED BY KNOW & 1
IS KNOW LESS THAN KSTRT
NO
YES
RESTORE SIGNS ON JCARD AND KCARD FIELDS
EXIT
KNOW IS REPLACED BY KSTRT & 1
JHASH IS REPLACED BY ZERO
KSTRT IS REPLACED BY J
JHASH IS REPLACED BY JHASH & JCARD&KSTRT¤
ICOMP IS REPLACED BY JCARD&KSTRT¤ - KCARD&KNOW¤
IS ICOMP EQUAL TO ZERO
NO
YES
KNOW IS REPLACED BY KNOW & 1
IS KSIGN * JSIGN LESS THAN ZERO
NO
YES
KSTRT IS REPLACED BY KSTRT & 1
IS KSTRT LESS THAN JLAST
YES
NO
IS JSIGN * KSIGN * JHASH NEGATIVE
NO
YES
ICOMP IS REPLACED BY 1
ICOMP IS REPLACED BY JSIGN * ICOMP

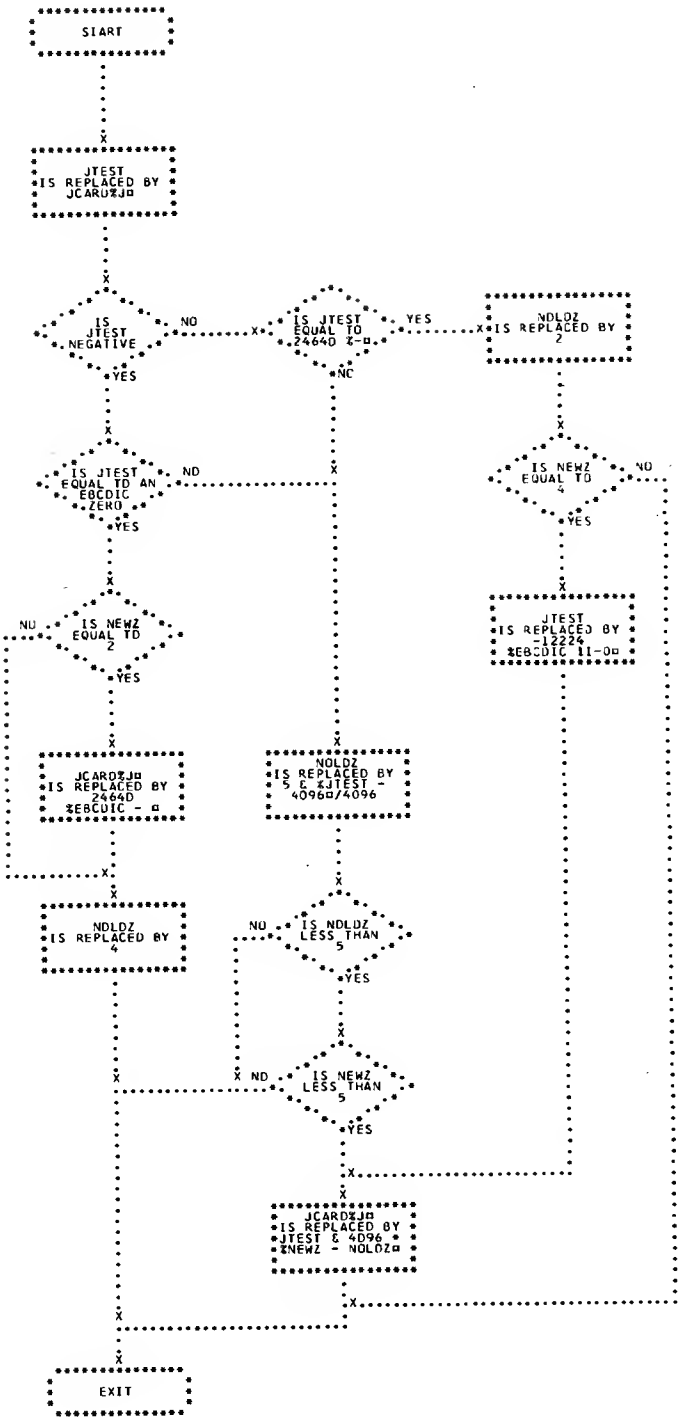

START
ANY INTERRUPTS PENDING
YES
NO
EXIT


START
JNOW
IS REPLACED BY
J
KNOW
IS REPLACED BY
K & JNOW - J
KCARD‡KNOW¤
IS REPLACED BY
JCARD‡JNOW¤
JNOW
IS REPLACED BY
JNOW & 1
IS JNOW
LESS THAN
JLAST
YES
NO
EXIT


START
KSTRT IS REPLACED BY K - JLAST & J -1
IS KSTRT GREATER THAN ZERO
NO
YES
NER IS REPLACED BY KLAST
CLEAR AND SAVE SIGNS ON JCARD AND KCARD FIELDS
FILL KCARD,KSTRT, K-1,0
JFRST IS REPLACED BY J
IS JCARD&JFRST□ POSITIVE
YES
NO
IS JFRST LESS THAN JLAST
YES
NO
JFRST IS REPLACED BY JFRST & 1
FILL KCARD,K, KLAST,0
J2
NSIGN JCARD,JLAST, JSIGN,JNOW
JNOW IS REPLACED BY JNOW & 1
EXIT
KM IS REPLACED BY K
MULT IS REPLACED BY KCARD&KM□
IS MULT GREATER THAN ZERO
NO
YES
KCARD&KM□ IS REPLACED BY ZERO
KNOW IS REPLACED BY KM & JFRST - JLAST
JNOW IS REPLACED BY JFRST
KCARD&KNOW□ IS REPLACED BY MULT * JCARD&JNOW□ & KCARD&KNOW□
KNOW IS REPLACED BY KNOW & 1
IS JNOW LESS THAN JLAST
YES
NO
KM IS REPLACED BY KM & 1
IS KM LESS THAN KLAST
YES
NO
CARRY KCARD,KSTRT, KLAST,KNOW
NSIGN KCARD,KLAST, JSIGN * KSIGN, KNOW
J2


START
JNOW
IS REPLACED BY
J
KNOW
IS REPLACED BY
K & JNOW - J
NCOMP
IS REPLACED BY
JCARD&JNOW¤/8
+ KCARD&KNOW¤/8
IS NCOMP
EQUAL TO
ZERO
NO
YES
JNOW
IS REPLACED BY
JNOW & 1
IS JNOW
GREATER THAN
JLAST
NO
YES
EXIT


START
NOLDS IS REPLACED BY 1
JTEST IS REPLACED BY JCARD$J¤
IS JTEST LESS THAN ZERO
NO
YES
NOLDS IS REPLACED BY -1
NEWS*JTEST IS COMPARED TO ZERO
LOW
EQL
HI
IS NEWS GREATER THAN ZERO
YES
NO
JTEST IS REPLACED BY -JTEST - 1
JCARD$J¤ IS REPLACED BY JTEST
EXIT


START
JTEST IS REPLACED BY JCARD¢J¤
IS JTEST NEGATIVE
NO
YES
IS JTEST EQUAL TO 2464D ¢-¤
YES
NO
NDLOZ IS REPLACED BY 2
IS JTEST EQUAL TO AN EBCDIC ZERO
NO
YES
IS NEWZ EQUAL TO 4
NO
YES
JTEST IS REPLACED BY -12224 ¢EBCDIC 11-0¤
NO
IS NEWZ EQUAL TO 2
YES
JCARD¢J¤ IS REPLACED BY 2464D ¢EBCDIC - ¤
NDLOZ IS REPLACED BY 5 ¢ ¢JTEST - 4096¤/4096
NDLOZ IS REPLACED BY 4
NO
IS NDLOZ LESS THAN 5
YES
X ND
IS NEWZ LESS THAN 5
YES
JCARD¢J¤ IS REPLACED BY JTEST ¢ 4D96 ¢NEWZ - NDLOZ¤
EXIT

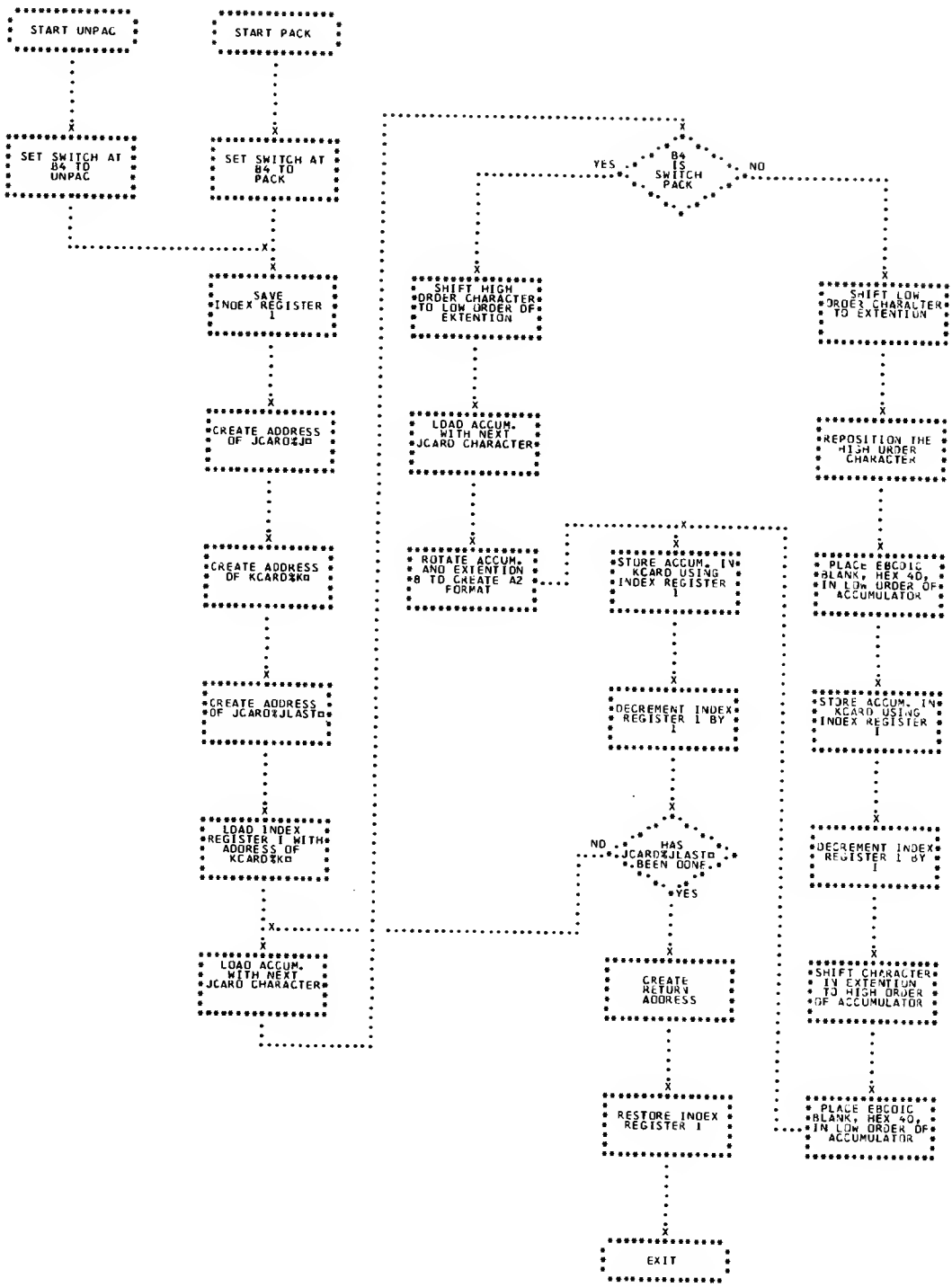
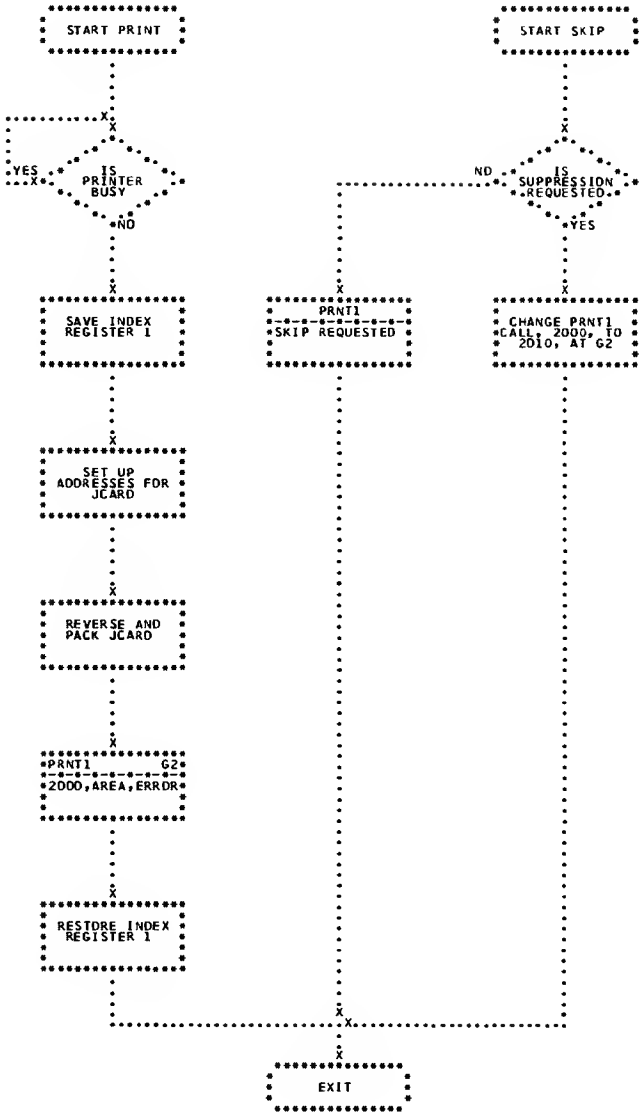
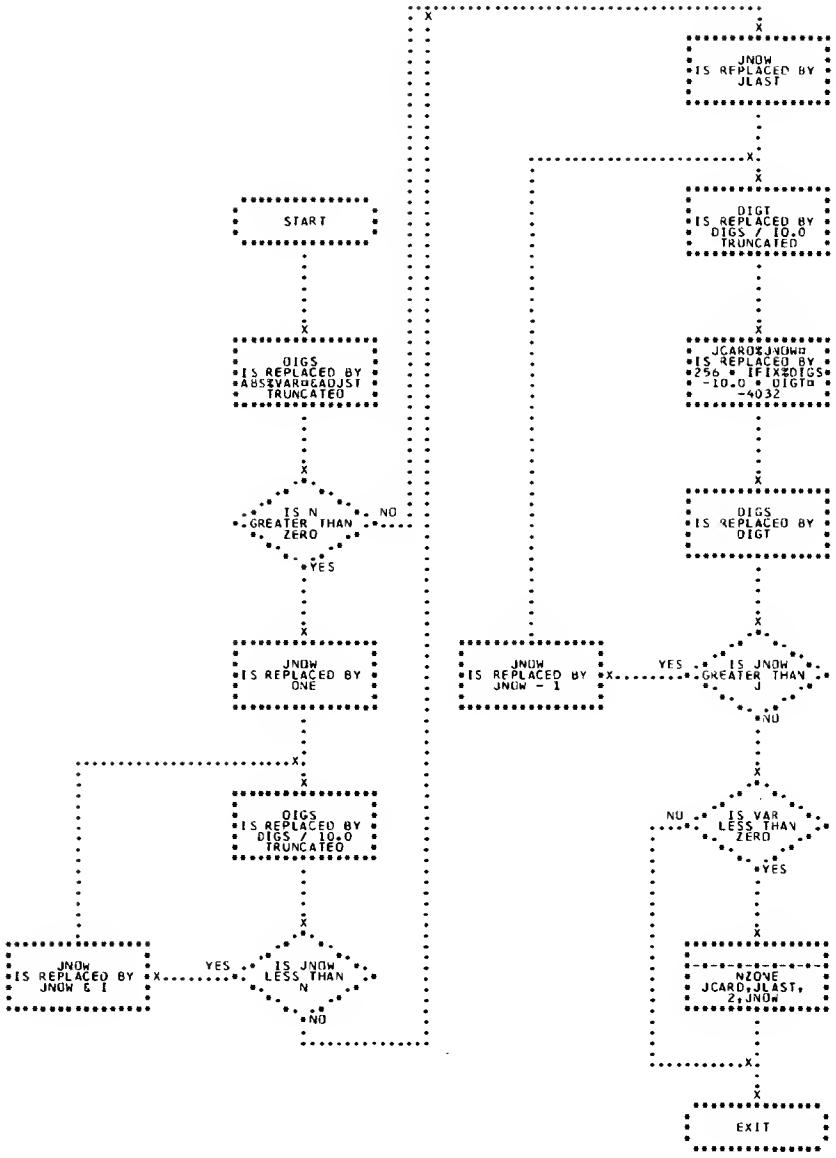
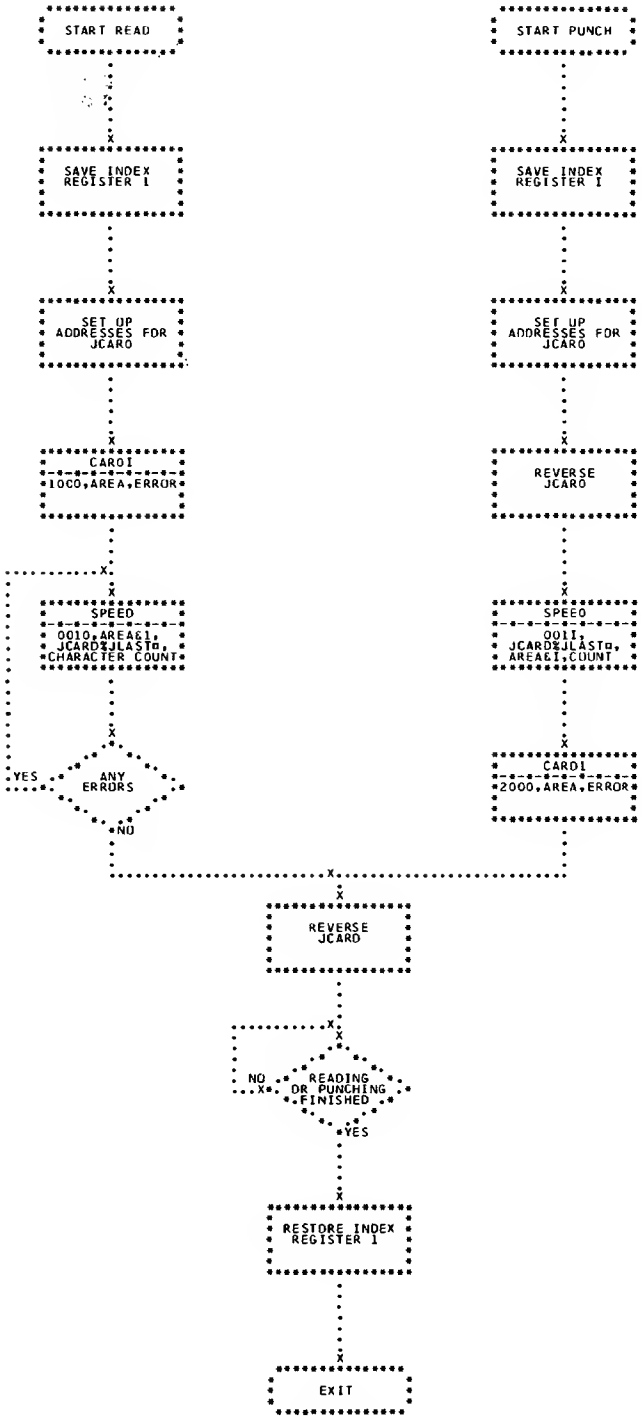

START UNPAC
START PACK
SET SWITCH AT B4 TO UNPAC
SET SWITCH AT B4 TO PACK
B4 IS SWITCH PACK
YES
NO
SAVE INDEX REGISTER 1
SHIFT HIGH ORDER CHARACTER TO LOW ORDER OF EXTENTION
SHIFT LOW ORDER CHARACTER TO EXTENTION
CREATE ADDRESS OF JCARD%JØ
LOAD ACCUM. WITH NEXT JCARD CHARACTER
REPOSITION THE HIGH ORDER CHARACTER
CREATE ADDRESS OF KCARD%KØ
ROTATE ACCUM. AND EXTENTION 8 TO CREATE A2 FORMAT
STORE ACCUM. IN KCARD USING INDEX REGISTER 1
PLACE EBCDIC BLANK, HEX 40, IN LOW ORDER OF ACCUMULATOR
CREATE ADDRESS OF JCARD%JLASTØ
DECREMENT INDEX REGISTER 1 BY 1
STORE ACCUM. IN KCARD USING INDEX REGISTER 1
LOAD INDEX REGISTER 1 WITH ADDRESS OF KCARD%KØ
HAS JCARD%JLASTØ BEEN DONE
NO
YES
DECREMENT INDEX REGISTER 1 BY 1
LOAD ACCUM. WITH NEXT JCARD CHARACTER
CREATE RETURN ADDRESS
SHIFT CHARACTER IN EXTENTION TO HIGH ORDER OF ACCUMULATOR
RESTORE INDEX REGISTER 1
PLACE EBCDIC BLANK, HEX 40, IN LOW ORDER OF ACCUMULATOR
EXIT

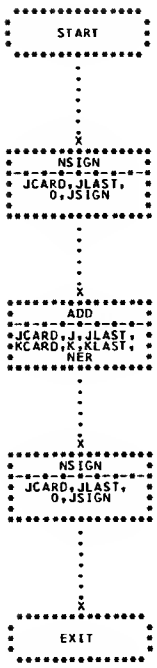
CHART PS 1130 COMMERCIAL PRINT/SKIP SUBROUTINE

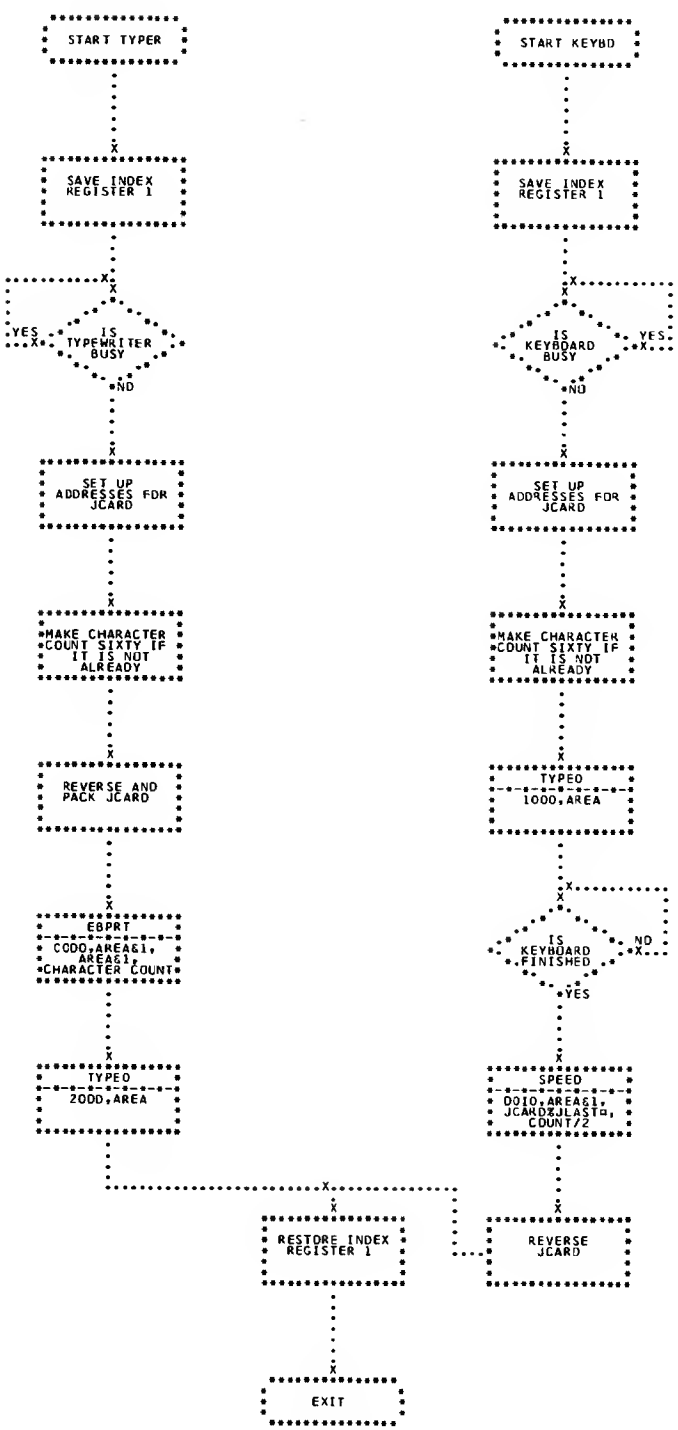


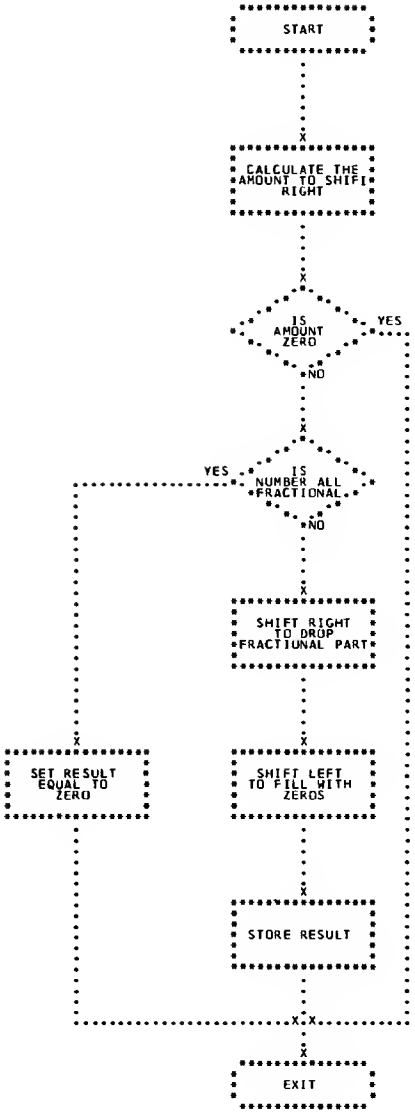

START
DIGS IS REPLACED BY ABS(VAR)&ADJST TRUNCATED
IS N GREATER THAN ZERO
NO
YES
JNOW IS REPLACED BY ONE
DIGS IS REPLACED BY DIGS / 10.0 TRUNCATED
JNOW IS REPLACED BY JNOW & I
YES
IS JNOW LESS THAN N
NO
JNOW IS REPLACED BY JNOW - 1
JNOW IS REPLACED BY JLAST
DIGT IS REPLACED BY DIGS / 10.0 TRUNCATED
JCARD(JNOW) IS REPLACED BY 256 * IFIX(DIGS -10.0 * DIGT) -4032
DIGS IS REPLACED BY DIGT
YES
IS JNOW GREATER THAN J
NO
NO
IS VAR LESS THAN ZERO
YES
NZONE JCARD,JLAST, 2,JNOW
EXIT


START READ
SAVE INDEX REGISTER 1
SET UP ADDRESSES FOR JCARD
CARDI
1000,AREA,ERROR
SPEED
0010,AREA&1,
JCARD%LAST¤,
CHARACTER COUNT
ANY ERRORS
YES
NO
START PUNCH
SAVE INDEX REGISTER 1
SET UP ADDRESSES FOR JCARD
REVERSE JCARD
SPEED
0011,
JCARD%LAST¤,
AREA&1,COUNT
CARDI
2000,AREA,ERROR
REVERSE JCARD
READING OR PUNCHING FINISHED
NO
YES
RESTORE INDEX REGISTER 1
EXIT


START
SELECT THE ALTERNATE STACKER
EXIT


START
NSIGN
JCARD,JLAST,
0,JSIGN
ADD
JCARD,J,JLAST,
KCARD,K,KLAST,
NER
NSIGN
JCARD,JLAST,
0,JSIGN
EXIT


START TYPER
SAVE INDEX REGISTER 1
IS TYPEWRITER BUSY
YES
NO
SET UP ADDRESSES FOR JCARD
MAKE CHARACTER COUNT SIXTY IF IT IS NOT ALREADY
REVERSE AND PACK JCARD
EBPRT
CODD,AREA&1,
AREA&1,
CHARACTER COUNT
TYPEO
20DD,AREA
START KEYBD
SAVE INDEX REGISTER 1
IS KEYBOARD BUSY
YES
NO
SET UP ADDRESSES FOR JCARD
MAKE CHARACTER COUNT SIXTY IF IT IS NOT ALREADY
TYPEO
10DD,AREA
IS KEYBOARD FINISHED
NO
YES
SPEED
DO10,AREA&1,
JCARD%LAST¤,
COUNT/2
REVERSE JCARD
RESTORE INDEX REGISTER 1
EXIT


START
CALCULATE THE AMOUNT TO SHIFT RIGHT
IS AMOUNT ZERO
YES
NO
IS NUMBER ALL FRACTIONAL
YES
NO
SHIFT RIGHT TO DROP FRACTIONAL PART
SET RESULT EQUAL TO ZERO
SHIFT LEFT TO FILL WITH ZEROS
STORE RESULT
EXIT

LISTINGS

```
// JOB                                                                  CSP00010
// FOR                                                                  CSP00020


                                                                        PAGE 01
** 1130 COMMERCIAL SUBROUTINE PACKAGE                                  CSP00030
* NAME NCOMP                                                           CSP00040
* ONE WORD INTEGERS                                                    CSP00050
* EXTENDED PRECISION                                                   CSP00060
* LIST ALL                                                             CSP00070


 1130 COMMERCIAL SUBROUTINE PACKAGE                                     PAGE 02
      FUNCTION NCOMP(JCARD,J,JLAST,KCARD,K)                             CSP00080
      DIMENSION JCARD(80), KCARD(80)                                    CSP00090
C-----COMPARE JCARD(J) WITH KCARD(K) THROUGH JCARD(JLAST).              CSP00100
C-----NCOMP=+0+ AS (JCARD-KCARD) IS -+0++                               CSP00110
C-----COLLATING SEQUENCE ASCENDING IS ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567CSP00120
C-----89 +(++$*)-/+(='+=                                               CSP00130
      DO 2 JNOW=J,JLAST                                                 CSP00140
      KNOW=K+JNOW-J                                                     CSP00150
      NCOMP=KCARD(KNOW)/8                                               CSP00160
      NCOMP=JCARD(JNOW)/8-NCOMP                                         CSP00170
      IF (NCOMP) 1,2,1                                                  CSP00180
2     CONTINUE                                                          CSP00190
1     RETURN                                                            CSP00200
      END                                                               CSP00210


 1130 COMMERCIAL SUBROUTINE PACKAGE                                     PAGE 03

VARIABLE ALLOCATIONS
 NCOMP=0000   JNOW =0001   KNOW =0002

STATEMENT ALLOCATIONS
 2     =003F  1     =0047

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC    SUBIN

INTEGER CONSTANTS
     8=0006

CORE REQUIREMENTS FOR NCOMP
 COMMON       0  VARIABLES       6  PROGRAM      70

END OF COMPILATION


// DUP                                                                  CSP00220

*STORE       WS  UA  NCOMP                                              CSP00230

2A45 0005


// FOR                                                                  CSP00240


                                                                        PAGE 01
** 1130 COMMERCIAL SUBROUTINE PACKAGE                                  CSP00250
* NAME MOVE                                                            CSP00260
* ONE WORD INTEGERS                                                    CSP00270
* EXTENDED PRECISION                                                   CSP00280
* LIST ALL                                                             CSP00290


 1130 COMMERCIAL SUBROUTINE PACKAGE                                     PAGE 02
      SUBROUTINE MOVE(JCARD,J,JLAST,KCARD,K)                            CSP00300
      DIMENSION JCARD(80), KCARD(80)                                    CSP00310
C-----MOVE JCARD(J) TO KCARD(K) THROUGH JCARD(JLAST).                   CSP00320
      DO 1 JNOW=J,JLAST                                                 CSP00330
      KNOW=K+JNOW-J                                                     CSP00340
1     KCARD(KNOW)=JCARD(JNOW)                                           CSP00350
      RETURN                                                            CSP00360
      END                                                               CSP00370
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 JNOW =0000  KNOW =0001

STATEMENT ALLOCATIONS
 1    =001E

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC    SUBIN

CORE REQUIREMENTS FOR MOVE
 COMMON       0  VARIABLES       4  PROGRAM      52

END OF COMPILATION



// DUP                                                                CSP00380

*STORE      WS  UA  MOVE                                               CSP00390

2A4A 0004



// FOR                                                                CSP00400



                                                                       PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP00410
* NAME EDIT                                                           CSP00420
* ONE WORD INTEGERS                                                   CSP00430
* EXTENDED PRECISION                                                  CSP00440
* LIST ALL                                                            CSP00450



  1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 02

       SUBROUTINE EDIT(JCARD,J,JLAST,KCARD,K,KLAST)                    CSP00460
       DIMENSION JCARD(80), KCARD(80)                                  CSP00470
C-----JCARD(J) THROUGH JCARD(JLAST) IS EDITED INTO EDIT FIELD.         CSP00480
C-----EDIT FIELD IS AT KCARD(K) THROUGH KCARD(KLAST).                  CSP00490
C-----CHECK FOR NEGATIVE FIELD. IF SO, CLEAR 11 ZONE.                  CSP00500
       CALL NZONE(JCARD,JLAST,4,NSIGN)                                 CSP00510
       NDUMP=16448                                                     CSP00520
       MONEY=16448                                                     CSP00530
       NZRSP=0                                                         CSP00540
C-----MAIN SCAN, INSERT CHARACTERS AND CHECK ZERO SUPPRESSION.         CSP00550
C-----BLANK=16448, 0=-4032, *=23616, $=23360, .=27456, -=24640, R=-9920.CSP00560
       KNOW=KLAST                                                      CSP00570
       JNOW=JLAST                                                      CSP00580
 17    KTEST=KCARD(KNDW)                                               CSP00590
       IF (KTEST) 33,34,34                                             CSP00600
 33    IF (KTEST+4032)  11,19,11                                       CSP00610
 34    IF (KTEST-16448) 13,20,13                                       CSP00620
 13    IF (KTEST-23616) 14,18,14                                       CSP00630
 14    IF (KTEST-23360) 11,28,11                                       CSP00640
 18    NDUMP=KTEST                                                     CSP00650
 28    MONEY=KTEST                                                     CSP00660
 19    NZRSP=KNOW                                                      CSP00670
 20    IF (JNOW-J) 11,26,26                                            CSP00680
 26    KTEST=JCARD(JNDW)                                               CSP00690
       KCARD(KNDW)=KTEST                                               CSP00700
       JNDW=JNOW-1                                                     CSP00710
       IF (NZRSP) 11,11,9                                              CSP00720
  9    IF (KTEST) 35,36,36                                             CSP00730
 35    IF (KTEST+4032)  25,11,25                                       CSP00740
 36    IF (KTEST-16448) 23,11,23                                       CSP00750
 23    IF (KTEST-27456) 25,11,25                                       CSP00760
 25    NZRSP=KNOW-1                                                    CSP00770
 11    KNOW=KNOW-1                                                     CSP00780
       IF (KNOW-K) 27,17,17                                            CSP00790
C-----RESTORE 11 ZONE IF FIELD WAS MINUS.                              CSP00800
 27    CALL NZONE(JCARD,JLAST,NSIGN,KTEST)                             CSP00810
C-----FILL FIELD WITH ASTERISKS IF OVERFLOW.                           CSP00820
       IF (JNDW-J) 29,21,21                                            CSP00830
 21    CALL FILL(KCARD,K,KLAST,23616)                                  CSP00840
C-----PAUSE 0001 HERE IF DESIRED FOR ERROR CORRECTION PURPOSES.        CSP00850
       GO TO 30                                                        CSP00860
C-----REMOVE CR OR - FROM EDIT IF POSITIVE FIELD.                      CSP00870
 29    IF (NSIGN-2) 2,3,2                                              CSP00880
  2    KTEST=KCARD(KLAST)                                              CSP00890
       IF (KTEST) 31,32,32                                             CSP00900
 32    IF (KTEST-24640) 3,5,3                                          CSP00910
 31    IF (KTEST+9920)  3,6,3                                          CSP00920
  6    KCARD(KLAST-1)=16448                                            CSP00930
  5    KCARD(KLAST)=16448                                              CSP00940
C-----ZERO SUPPRESSION, NDUMP IS SUPPRESSION CODE. NZRSP IS END.       CSP00950
  3    IF (NZRSP) 30,30,22                                             CSP00960
 22    CALL FILL(KCARD,K,NZRSP,NDUMP)                                  CSP00970
C-----INSERT FLOATING DOLLAR SIGN OR ASTERISK OR BLANK.                 CSP00980
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

      KCARO(NZRSP)=MONEY                                               CSP00990
30    RETURN                                                           CSP01000
      END                                                              CSP01010



 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 04

VARIABLE ALLOCATIONS
 NSIGN=0000  NOUMP=0001  MONEY=0002  NZRSP=0003  KNOW =0004  JNOW =0005  KTEST=0006

STATEMENT ALLOCATIONS
 17   =0050  33   =0050  34   =0065  13   =0068  14   =0071  18   =0079  28   =0070  19   =0081  20   =0085  26   =0088
 9    =00A7  35   =00AB  36   =0083  23   =0089  25   =008F  11   =00C5  27   =0001  21   =0000  29   =00E5  2    =00EB
 32   =00F8  31   =0100  6    =0106  5    =010F  3    =0118  22   =011C  30   =0128

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NZONE   FILL    SUBSC   SUBIN

INTEGER CONSTANTS
     4=000A  16448=0008       0=000C   4032=0000  23616=000E  23360=000F       1=0010  27456=0D11       2=0012  24640=0013
  9920=0014

CORE REQUIREMENTS FOR EDIT
 COMMON       0  VARIABLES     10  PROGRAM    292

END OF COMPILATION



// OUP                                                                 CSP01020

*STORE      WS  UA  EDIT                                               CSP01030

2A4E 0012



// FOR                                                                 CSP01040



                                                                       PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP01050
* NAME GET                                                            CSP01060
* ONE WORD INTEGERS                                                   CSP01070
* EXTENDED PRECISION                                                  CSP01080
* LIST ALL                                                            CSP01090



 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 02

      FUNCTION GET(JCARD,J,JLAST,SHIFT)                                CSP01100
      DIMENSION JCARD(80)                                              CSP01110
C-----GET FIELD FROM JCARO(J) THROUGH JCARO(JLAST).                    CSP01120
C-----SHIFT = A CONSTANT TO SHIFT DECIMAL PT FROM WHOLE NUMBER FORM.   CSP01130
C-----CHECK SIGN. CLEAR 11 ZONE IF NEGATIVE.                           CSP01140
      CALL NZONE(JCARD,JLAST,4,NSIGN)                                  CSP01150
      GET=0.                                                           CSP01160
C-----START MAIN SCAN. GET DIGITS.                                     CSP01170
      DO 3 JNOW=J,JLAST                                                CSP01180
      JTEST=JCARO(JNOW)                                                CSP01190
C-----CHECK FOR BLANK OR NON-NUMERIC CHARACTER. GET=0. IF NON-NUMERIC.  CSP01200
      IF (JTEST) 4,2,2                                                 CSP01210
2     IF (JTEST-16448) 6,5,6                                           CSP01220
5     JTEST=-4032                                                      CSP01230
4     IF (JTEST+4032) 6,3,3                                            CSP01240
C-----BLANK=16448. 0=-4032.                                            CSP01250
3     GET=10.*GET+FLOAT((JTEST+4032)/256)                              CSP01260
C-----SHIFT DECIMAL POINT.                                             CSP01270
      GET=SHIFT*GET                                                    CSP01280
C-----CHECK SIGN. RESTORE 11 ZONE IF NEEDED.                           CSP01290
      CALL NZONE(JCARO,JLAST,NSIGN,JTEST)                              CSP01300
      IF (NSIGN-2) 7,11,7                                              CSP01310
11    GET=-GET                                                         CSP01320
7     RETURN                                                           CSP01330
C-----SET GET=0. IF NON-NUMERIC CHARACTER.                             CSP01340
6     GET=0.                                                           CSP01350
C-----PAUSE 0003 HERE IF DESIRED FOR ERROR CORRECTION PURPOSES.        CSP01360
      GO TO 7                                                          CSP01370
      END                                                              CSP01380
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 GET  =0000  NSIGN=0009  JNOW =000A  JTEST=000B

STATEMENT ALLOCATIONS
 2    =0043  5    =0049  4    =004E  3    =0054  11   =0084  7    =0089  6    =008D

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NZONE   EADD    EMPY    ELD     ESTO    FLOAT   SUBSC   SNR     SUBIN

REAL CONSTANTS
   .000000000E 00=000E     .100000000E 02=0011

INTEGER CONSTANTS
     4=0014  16448=0015   4032=0016    256=0017      2=0018

CORE REQUIREMENTS FOR GET
 COMMON      0  VARIABLES     14  PROGRAM    134

END OF COMPILATION



// DUP                                                                CSP01390

*STORE      WS  UA  GET                                                CSP01400

2A60 000A



// ASM                                                                CSP01410
** WHOLE NUMBER SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE  (IO) CSP01420
* NAME WHOLE                                                      (IO) CSP01430
* LIST (1132 PRINTER)                                                CSP01440



                                                                     PAGE  1

0006    262164C5          ENT     WHOLE     SUBROUTINE ENTRY POINT     CSP01450
                  # X=WHOLE(Y). WITH Y IN FAC TO START                  CSP01460
                  # X IN FAC BECOMES THE INTEGRAL PART OF Y.           CSP01470
0000 0  0000      OBL1  DC      0         OBL CONSTANT OF 1            CSP01480
0001 0  0001            DC      1         REST OF OBL1 CONSTANT        CSP01490
001F              MANT  EQU     31        MANTISSA LENGTH              CSP01500
0002 0  009F      C159  DC      128+MANT  EXPONENT OF FULL INTEGER     CSP01510
0003 0  001F      C31   DC      MANT      MANTISSA LENGTH              CSP01520
0004 0  189F      SRT   SRT     MANT      SRT MANTISSA LENGTH          CSP01530
0005 0  0800      H0800 DC      /0800     DIFF BETWEEN SRT AND SLT     CSP01540
0006 0  0000      WHOLE DC      0         ARGUMENT ADDRESS HERE        CSP01550
0007 0  C0FA            LD      C159      EXP OF FULL INTEGER          CSP01560
0008 0  9370            S     3 125       SUBTRACT EXP OF Y            CSP01570
0009 01 4C2B001A        BSC  L  ONE+*+2   BRANCH IF ALL INTEGER         CSP01580
000B 0  90F7            S       C31       SUBTRACT MANTISSA LENGTH     CSP0159C
000C 01 4C10001E        BSC  L  FRACT+-   BRANCH IF ALL FRACTIONAL     CSP0160C
000E 0  80F5            A       SRT       CREATE RIGHT SHIFT           CSP01610
000F 0  0005            STO     RIGHT     STORE  RIGHT SHIFT           CSP01620
0010 0  90F4            S       H0800     CREATE  LEFT SHIFT           CSP01630
0011 0  0006            STO     LEFT      STORE   LEFT SHIFT           CSP01640
0012 0  CB7E            LDD   3 126       PICK UP MANTISSA             CSP01650
0013 0  4828            BSC     +Z        CHECK FOR NEGATIVE MANTISA   CSP01660
0014 0  98E8            SD      OBL1      SUBTRACT 1 IF NEGATIVE       CSP01670
0015 0  1880      RIGHT SRT     0         RIGHT SHIFT                  CSP01680
0016 0  4828            BSC     +Z        CHECK FOR NEGATIVE MANTISA   CSP01690
0017 0  88E8            AD      OBL1      ADD 1 IF NEGATIVE            CSP01700
0018 0  1080      LEFT  SLT     0         LEFT SHIFT                   CSP01710
0019 0  0B7E      STORE STO   3 126       STORE MANTISSA               CSP01720
001A 01 74010006  ONE   MDX  L  WHOLE+1  CREATE RETURN ADDRESS         CSP01730
001C 01 4C800006        BSC  I  WHOLE     RETURN TO CALLING PROGRAM    CSP01740
001E 0  10E0      FRACT SLC     32        ZERO ACC AND EXT             CSP01750
001F 0  0370            STO   3 125       ZERO THE EXPONENT            CSP01760
0020 0  70FB            MDX     STORE     ZERO THE MANTISSA            CSP01770
0022                    END               END OF WHOLE SUBROUTINE      CSP01780

    NO ERRORS IN ABOVE ASSEMBLY.



// DUP                                                                CSP01790

*STORE      WS  UA  WHOLE                                              CSP01800

2A6A 0003



// FOR                                                                CSP01810



                                                                     PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP01820
*NAME PUT                                                            CSP01830
*ONE WORD INTEGERS                                                   CSP01840
*EXTENDED PRECISION                                                  CSP01850
*LIST ALL                                                            CSP01860
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 02

      SUBROUTINE PUT(JCARD,J,JLAST,VAR,AOJST,N)                        CSP01870
      DIMENSION JCARO(10)                                              CSP01880
C-----PUT VAR INTO JCARO(J) THROUGH JCARD(JLAST).                      CSP01890
C-----ADJST = A NUMBER TO HALF ADJUST THE VARIABLE VAR.                CSP01900
C-----N = THE NUMBER OF POSITIONS THE DECIMAL POINT SHOULD BE MOVED LEFTCSP01910
      DIGS=WHOLE(ABS(VAR)+AOJST)                                       C5P01920
      IF(N)   3,3,1                                                    CSP01930
1     DO 2 JNOW=1,N                                                    CSP01940
2     OIGS=WHOLE(OIGS*0.1)                                             CSP01950
C-----PUT OIGITS IN FIELO                                              CSP01960
3     JNOW=JLAST                                                       CSP01970
4     DIGT=WHOLE(DIGS*0.1)                                             CSP01980
      JCARD(JNOW)=256*IFIX(DIGS-10.0*OIGT)-4032                        CSP01990
      OIGS=OIGT                                                        CSP02000
      IF(JNOW-J)   6,6,5                                               CSP02010
5     JNOW=JNOW-1                                                      CSP02020
      GO TO 4                                                          CSP02030
C-----PUT 11 PUNCH OVER LOW ORDER DIGIT IF NEGATIVE.                   CSP02040
6     IF(VAR)  7,8,8                                                   CSP02050
7     CALL NZONE(JCARD,JLAST,2,JNOW)                                   CSP02060
8     RETURN                                                           CSP02070
      END                                                              CSP02080


 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 OIGS =0000  DIGT =0003  JNOW =0009

STATEMENT ALLOCATIONS
 1    =0039  2    =0030  3    =0050  4    =0054  5    =0082  6    =008A  7    =008F  8    =0095

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 WHOLE   EABS    NZONE   EAOO    EMPY    ELD     ESTO    ESBR    IFIX    SUB5C   SUBIN

REAL CONSTANTS
   .100000000E 00=000C     .100000000E 02=000F

INTEGER CONSTANTS
     1=0012    256=0013   4032=0014      2=0015

CORE REQUIREMENTS FOR PUT
 COMMON       0  VARIABLES     12  PROGRAM    140

END OF COMPILATION



// DUP                                                                 CSP02090

*STORE      WS  UA  PUT                                                 CSP02100

2A6D 000A



// FOR                                                                 CSP02110



                                                                       PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                  CSP02120
* NAME NZONE                                                           CSP02130
* ONE WORD INTEGERS                                                    CSP02140
* EXTENDED PRECISION                                                   CSP02150
* LIST ALL                                                             CSP02160



 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 02

      SUBROUTINE NZONE(JCARO,J,NEWZ,NOLDZ)                             CSP02170
      DIMENSION JCARO(80)                                              CSP02180
C-----SPECIAL CHARACTER,  JCARD(J) IS THEN ZONED 12 ZONE, 11 ZONE,     CSP02190
C-----NOLDZ=1,2,3,4, OR MORE AS JCARD(J) WAS A-I, J-R, S-Z, 0-9, OR A  CSP02200
C-----0 ZONE, NO ZONE, OR LEFT ALONE AS NEWZ=1,2,3,4, OR MORE,         CSP02210
C-----JCARD(J) IS ALWAYS LEFT ALONE IF JCARD(J) WAS A SPECIAL CHARACTER,CSP02220
C-----ZERO CHANGED TO 11 ZONE IS -, - CHANGED TO NO ZONE IS ZERO.      CSP02230
C-----ZERO CODE IS =4032, - CODE IS 24640,                             CSP02240
      JTEST=JCARD(J)                                                   CSP02250
      IF (JTEST) 4,5,5                                                 CSP02260
    4 IF (JTEST+4032) 8,6,8                                            CSP02270
    6 IF (NEWZ-2) 12,7,12                                              CSP02280
    7 JCARD(J)=24640                                                   CSP02290
   12 NOLDZ=4                                                          CSP02300
      GO TO 2                                                          CSP02310
    5 IF (JTEST-24640) 8,10,8                                          CSP02320
   10 NOLDZ=2                                                          CSP02330
      IF (NEWZ-4) 2,9,2                                                CSP02340
    9 JTEST=-12224                                                     CSP02350
    3 JCARD(J)=JTEST+4096*(NEWZ-NOLDZ)                                 CSP02360
    2 RETURN                                                           CSP02370
    8 NOLDZ=5+(JTEST-4096)/4096                                        CSP02380
      IF (NOLDZ-5) 1,2,2                                               CSP02390
    1 IF (NEWZ-5) 3,2,2                                                CSP02400
      END                                                              CSP02410
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 JTEST=0000

STATEMENT ALLOCATIONS
 4    =002C  6    =0032  7    =0038  12   =0041  5    =0047  10   =004D  9    =0057  3    =005C  2    =006C  8    =006E
 1    =007F

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC   SUBIN

INTEGER CONSTANTS
  4032=0002      2=0003  24640=0004      4=0005  12224=0006   4096=0007      5=0008

CORE REQUIREMENTS FOR N2ONE
 COMMON       0  VARIABLES      2  PROGRAM    134

END OF COMPILATION



// DUP                                                                CSP02420

*STORE      WS  UA  NZONE                                             CSP02430

2A77 0009



// FOR                                                                CSP02440



                                                                      PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP02450
* NAME FILL                                                           CSP02460
* ONE WORD INTEGERS                                                   CSP02470
* EXTENDED PRECISION                                                  CSP02480
* LIST ALL                                                            CSP02490



 1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 02

      SUBROUTINE FILL(JCARD,J,JLAST,NCH)                              CSP02500
      DIMENSION JCARD(80)                                             CSP02510
C-----FILL JCARD(J) THROUGH JCARD(JLAST) WITH NCH.                    CSP02520
      DO 1 JNOW=J,JLAST                                               CSP02530
1     JCARD(JNOW)=NCH                                                 CSP02540
      RETURN                                                          CSP02550
      END                                                             CSP02560



 1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 03

VARIABLE ALLOCATIONS
 JNOW =0000

STATEMENT ALLOCATIONS
 1    =0011

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC   SUBIN

CORE REQUIREMENTS FOR FILL
 COMMON       0  VARIABLES      2  PROGRAM     34

END OF COMPILATION



// DUP                                                                CSP02570

*STORE      WS  UA  FILL                                              CSP02580

2A80 0003



// ASM                                                                CSP02590
** STACKER SELECT SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE(10) CSP02600
* NAME STACK                                                    (10) CSP02610
* LIST (1132 PRINTER)                                                CSP02620
```

```
                                                                        PAGE  1
0002    228C10D2          ENT     STACK                                 CSP02630
0000 0  0000        IOCC  DC      0                                     CSP02640
0001 0  1480              DC      /1480                                 CSP02650
0002 0  0000        STACK DC      *-*                                   CSP02660
0003 0  0BFC              XIO     IOCC          SELECT STACKER          CSP02670
0004 01 4C800002          BSC  I  STACK         RETURN                  CSP02680
0006                      END                                           CSP02690

    NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                  CSP02700

*STORE      WS  UA  STACK                                               CSP02710

2A83 0002


// FOR                                                                  CSP02720


                                                                        PAGE 01
** 1130 COMMERCIAL SUBROUTINE PACKAGE                                   CSP02730
* NAME ADD                                                             CSP02740
* ONE WORD INTEGERS                                                    CSP02750
* EXTENDED PRECISION                                                   CSP02760
* LIST ALL                                                             CSP02770


 1130 COMMERCIAL SUBROUTINE PACKAGE                                     PAGE 02

      SUBROUTINE ADD(JCARD,J,JLAST,KCARD,K,KLAST,NER)                   CSP02780
      DIMENSION JCARD(80), KCARD(80)                                    CSP02790
C-----ADD FIELD AT JCARD TO FIELD AT KCARD, SUBTRACTING IF UNLIKE SIGNS.CSP02800
C-----SET NER=KLAST IF OVERFLOW OCCURS.                                 CSP02810
C-----CLEAR SIGNS AND ADD.                                              CSP02820
      CALL NSIGN(JCARD,JLAST,1,JSIGN)                                   CSP02830
      CALL NSIGN(KCARD,KLAST,1,KSIGN)                                   CSP02840
      LSIGN=JSIGN*KSIGN                                                 CSP02850
      KNOW=KLAST+J-JLAST                                                CSP02860
      IF (KNOW-K) 8,7,7                                                 CSP02870
7     DO 6 JNOW=J,JLAST                                                 CSP02880
      KCARD(KNOW)=LSIGN*JCARD(JNOW)+KCARD(KNOW)                         CSP02890
6     KNOW=KNOW+1                                                       CSP02900
5     CALL CARRY(KCARD,K,KLAST,KNOW)                                    CSP02910
      IF (KNOW) 1,2,3                                                   CSP02920
C-----COMPLEMENT AND CHANGE SIGN OF KCARD IF CARRY IS NEGATIVE.         CSP02930
1     KCARD(KLAST)=KCARD(KLAST)+KNOW                                    CSP02940
      DO 4 KNOW=K,KLAST                                                 CSP02950
4     KCARD(KNOW)=9-KCARD(KNOW)                                         CSP02960
      KSIGN=-KSIGN                                                      CSP02970
      GO TO 5                                                           CSP02980
C-----OVERFLOW HAS OCCURRED IF CARRY IS POSITIVE.                       CSP02990
3     CALL FILL(KCARD,K,KLAST,9)                                        CSP03000
8     NER=KLAST                                                         CSP03010
C-----RESTORE SIGNS AND RETURN.                                         CSP03020
2     CALL NSIGN(KCARD,KLAST,KSIGN,KNOW)                                CSP03030
      CALL NSIGN(JCARD,JLAST,JSIGN,JNOW)                                CSP03040
      RETURN                                                            CSP03050
      END                                                               CSP03060


 1130 COMMERCIAL SUBROUTINE PACKAGE                                     PAGE 03

VARIABLE ALLOCATIONS
 JSIGN=0000  KSIGN=0001  LSIGN=0002  KNOW =0003  JNOW =0004

STATEMENT ALLOCATIONS
 7    =0055  6    =006E  5    =007C  1    =00BB  4    =0097  3    =00B1  8    =00B7  2    =00BB

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NSIGN   CARRY   FILL    SUBSC   SUBIN

INTEGER CONSTANTS
     1=0008      9=0009

CORE REQUIREMENTS FOR ADD
 COMMON      0  VARIABLES      8  PROGRAM    194

END OF COMPILATION


// DUP                                                                  CSP03070

*STORE      WS  UA  ADD                                                 CSP03080

2AB5 000C


// FOR                                                                  CSP03090
```

```
                                                                      PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP03100
* NAME SUB                                                           CSP03110
* ONE WORD INTEGERS                                                  CSP03120
* EXTENDED PRECISION                                                 CSP03130
* LIST ALL                                                           CSP03140



 1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 02

      SUBROUTINE SUB(JCARD,J,JLAST,KCARD,K,KLAST,NER)                 CSP03150
      DIMENSION JCARD(80), KCARD(80)                                  CSP03160
C-----SUBTRACT FIELD AT JCARD FROM FIELD AT KCARD.                    CSP03170
C-----SET NER=KLAST IF OVERFLOW.                                      CSP03180
C-----CHANGE THE SIGN OF JCARD. THEN ADD.                             CSP03190
      CALL NSIGN(JCARD,JLAST,0,JSIGN)                                 CSP03200
      CALL ADD(JCARD,J,JLAST,KCARD,K,KLAST,NER)                       CSP03210
      CALL NSIGN(JCARD,JLAST,0,JSIGN)                                 CSP03220
      RETURN                                                          CSP03230
      END                                                             CSP03240



 1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 03

VARIABLE ALLOCATIONS
 JSIGN=0000

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NSIGN   ADD     SUBIN

INTEGER CONSTANTS
     0=0002

CORE REQUIREMENTS FOR SUB
 COMMON      0  VARIABLES      2  PROGRAM     46

END OF COMPILATION



// DUP                                                              CSP03250

*STORE      WS  UA  SUB                                             CSP03260

2A91 0004



// FOR                                                              CSP03270



                                                                      PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP03280
* NAME MPY                                                           CSP03290
* ONE WORD INTEGERS                                                  CSP03300
* EXTENDED PRECISION                                                 CSP03310
* LIST ALL                                                           CSP03320



 1130 COMMERCIAL SUBROUTINE PACKAGE                                   PAGE 02

      SUBROUTINE MPY(JCARD,J,JLAST,KCARD,K,KLAST,NER)                 CSP03330
      DIMENSION JCARD(80), KCARD(80)                                  CSP03340
C-----MULTIPLY FIELD AT JCARD TIMES FIELD AT KCARD, PRODUCT REPLACES  CSP03350
C-----FIELD AT KCARD, WHICH IS EXTENDED TO THE LEFT.                  CSP03360
      KSTRT=K+J-JLAST-1                                               CSP03370
      IF(KSTRT)              6,6,7                                    CSP03380
6     NER=KLAST                                                       CSP03390
      GO TO 9                                                         CSP03400
C-----SAVE SIGNS.                                                     CSP03410
7     CALL NSIGN(JCARD,JLAST,1,JSIGN)                                 CSP03420
      CALL NSIGN(KCARD,KLAST,1,KSIGN)                                 CSP03430
C-----FILL KCARD EXTENSION WITH ZEROS.                                CSP03440
      CALL FILL(KCARD,KSTRT,K-1,0)                                    CSP03450
C-----PASS OVER LEADING ZEROS TO FIND JFRST.                          CSP03460
      DO 3 JFRST=J,JLAST                                              CSP03470
      IF (JCARD(JFRST)) 3,3,4                                         CSP03480
3     CONTINUE                                                        CSP03490
C-----SET KCARD TO ZEROS IF ALL ZEROS.                                CSP03500
      CALL FILL(KCARD,K,KLAST,0)                                      CSP03510
      GO TO 8                                                         CSP03520
C-----MAIN MULTIPLICATION LOOP FOLLOWS.                               CSP03530
4     DO 1 KM=K,KLAST                                                 CSP03540
      MULT=KCARD(KM)                                                  CSP03550
      IF (MULT) 1,1,2                                                 CSP03560
2     KCARD(KM)=0                                                     CSP03570
      KNOW=KM+JFRST-JLAST                                             CSP03580
      DO 5 JNOW=JFRST,JLAST                                           CSP03590
      KCARD(KNOW)=MULT*JCARD(JNOW)+KCARD(KNOW)                        CSP03600
5     KNOW=KNOW+1                                                     CSP03610
1     CONTINUE                                                        CSP03620
      CALL CARRY(KCARD,KSTRT,KLAST,KNOW)                              CSP03630
C-----RESTORE SIGNS AND RETURN.                                       CSP03640
      CALL NSIGN(KCARD,KLAST,JSIGN*KSIGN,KNOW)                        CSP03650
8     CALL NSIGN(JCARD,JLAST,JSIGN,JNOW)                              CSP03660
9     RETURN                                                          CSP03670
      END                                                             CSP03680
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 KSTRT=0003  JSIGN=0004  KSIGN=0005  JFRST=0006  KM   =0007  MULT =0008  KNOW =0009  JNOW =000A

STATEMENT ALLOCATIONS
 6    =0048  7    =004E  3    =0073  4    =0083  2    =0094  5    =00BE  1    =00CC  8    =00E7  9    =00ED

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NSIGN   FILL    CARRY   SUBSC   SUBIN

INTEGER CONSTANTS
     I=000E      0=000F

CORE REQUIREMENTS FOR MPY
 COMMON      0  VARIABLES     14  PROGRAM    226

END OF COMPILATION



// DUP                                                                CSP03690

*STORE      WS  UA  MPY                                                CSP03700

2A95 000E



// FOR                                                                CSP03710



                                                                       PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP03720
* NAME DIV                                                            CSP03730
* ONE WORD INTEGERS                                                   CSP03740
* EXTENDED PRECISION                                                  CSP03750
* LIST ALL                                                            CSP03760



 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 02

      SUBROUTINE DIV(JCARD,J,JLAST,KCARD,K,KLAST,NER)                  CSP03770
      DIMENSION JCARD(80), KCARD(80)                                   CSP03780
C-----DIVIDE FIELD AT KCARD BY FIELD AT JCARD. QUOTIENT REPLACES FIELD  CSP03790
C-----AT KCARD, WHICH IS EXTENDED TO THE LEFT.                         CSP03800
C-----REMAINDER IS IN LOW ORDER DIGITS OF KCARD.                       CSP03810
C-----SET NER=KLAST IF DIVISION BY ZERO ATTEMPTED.                     CSP03820
C-----SAVE SIGNS.                                                      CSP03830
      CALL NSIGN(JCARD,JLAST,1,JSIGN)                                  CSP03840
      CALL NSIGN(KCARD,KLAST,1,KSIGN)                                  CSP03850
C-----FILL KCARD EXTENSION WITH ZEROS.                                 CSP03860
      JSPAN=JLAST-J+1                                                  CSP03870
      KSTRT=K-1                                                        CSP03880
      KLOW=K-JSPAN                                                     CSP03890
      IF (KLAST-KSTRT-JSPAN) 9,10,10                                   CSP03900
10    IF(KLOW)   9,9,11                                                CSP03910
11    CALL FILL(KCARD,KLOW,KSTRT,0)                                    CSP03920
C-----PASS OVER LEADING ZEROS TO FIND JFRST.                           CSP03930
      DO 3 JFRST=J,JLAST                                               CSP03940
      IF (JCARD(JFRST)) 3,3,4                                          CSP03950
3     CONTINUE                                                         CSP03960
C-----DIVISION BY ZERO WAS ATTEMPTED.                                  CSP03970
9     NER=KLAST                                                        CSP03980
      GO TO 8                                                          CSP03990
C-----FIND TRIAL DIVISOR AND DIVIDE.                                   CSP04000
4     JHIGH=JCARD(JFRST)                                               CSP04010
      KPUT=KLOW+JLAST-JFRST                                            CSP04020
      KSTOP=KLAST+JFRST-JLAST-1                                        CSP04030
      DO 1 KM=KSTRT,KSTOP                                              CSP04040
      MULT=(10*KCARD(KM)+KCARD(KM+1))/JHIGH                            CSP04050
      NOUO=MULT                                                        CSP04060
      IF (MULT) 7,7,2                                                  CSP04070
2     KNOW=KM+1                                                        CSP04080
      DO 5 JNOW=JFRST,JLAST                                            CSP04090
      KCARD(KNOW)=KCARD(KNOW)-MULT*JCARD(JNOW)                         CSP04100
5     KNOW=KNOW+1                                                      CSP04110
      CALL CARRY(KCARD,KM,KNOW-1,KNOW)                                 CSP04120
C-----ADD BACK DIVISOR IF OVERDRAWN.                                   CSP04130
      IF (KNOW) 6,7,7                                                  CSP04140
6     KCARD(KM)=KCARD(KM)+10*KNOW                                      CSP04150
      MULT=-1                                                          CSP04160
      NOUO=NOUO-1                                                      CSP04170
      GO TO 2                                                          CSP04180
C-----STORE QUOTIENT DIGIT.                                            CSP04190
7     KCARD(KPUT)=NOUO                                                 CSP04200
1     KPUT=KPUT+1                                                      CSP04210
C-----RESTORE SIGNS. REMAINDER GETS SIGN OF DIVIDEND.                  CSP04220
      CALL NSIGN(KCARD,KLAST-JSPAN,JSIGN*KSIGN,KNOW)                   CSP04230
8     CALL NSIGN(JCARD,JLAST,JSIGN,JNOW)                               CSP04240
      CALL NSIGN(KCARD,KLAST,KSIGN,KNOW)                               CSP04250
      RETURN                                                           CSP04260
      END                                                              CSP04270
```

```
1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 JSIGN=0006  KSIGN=0007  JSPAN=0008  KSTRT=0009  KLOW =000A  JFRST=000B  JHIGH=000C  KPUT =0000  KSTOP=000E  KM   =000F
 MULT =0010  NOUO =0011  KNOW =0012  JNOW =0013

STATEMENT ALLOCATIONS
 10   =006E  11   =0072  3    =0085  9    =0080  4    =0093  2    =0000  5    =00F1  6    =010F  7    =012A  1    =0133
 8    =0154

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NSIGN   FILL    CARRY   SUBSC   SUBIN

INTEGER CONSTANTS
     1=0016      0=0017     10=0018

CORE REQUIREMENTS FOR DIV
 COMMON       0  VARIABLES     22  PROGRAM     332

END OF COMPILATION



// DUP                                                                CSP04280

*STORE      WS  UA  DIV                                               CSP04290

2AA3 0015



// FOR                                                                CSP04300



                                                                      PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP04310
* NAME ICOMP                                                          CSP04320
* ONE WORD INTEGERS                                                   CSP04330
* EXTENDED PRECISION                                                  CSP04340
* LIST ALL                                                            CSP04350



  1130 COMMERCIAL SUBROUTINE PACKAGE                                  PAGE 02

       FUNCTION ICOMP(JCARD,J,JLAST,KCARD,K,KLAST)                    CSP04360
       DIMENSION JCARD(80), KCARD(80)                                 CSP04370
C-----COMPARE INTEGER FIELD AT KCARD AGAINST FIELD AT JCARD.          CSP04380
C-----ICOMP=+0+ AS (JCARD-KCARD) IS =+0+.                             CSP04390
       CALL NSIGN(JCARD,JLAST,1,JSIGN)                                CSP04400
       CALL NSIGN(KCARD,KLAST,1,KSIGN)                                CSP04410
       KSTRT=KLAST+J-JLAST-1                                          CSP04420
       IF (K-KSTRT) 5,5,4                                             CSP04430
5      ICOMP=-KSIGN                                                   CSP04440
       DO 6 KNOW=K,KSTRT                                              CSP04450
       IF (KCARD(KNOW)) 6,6,3                                         CSP04460
6      CONTINUE                                                       CSP04470
4      KNOW=KSTRT+1                                                   CSP04480
       JHASH=0                                                        CSP04490
       DO 2 KSTRT=J,JLAST                                             CSP04500
       JHASH=JHASH+JCARD(KSTRT)                                       CSP04510
       ICOMP=JCARD(KSTRT)-KCARD(KNOW)                                 CSP04520
       IF (ICOMP) 1,2,1                                               CSP04530
2      KNOW=KNOW+1                                                    CSP04540
       IF (JSIGN*KSIGN*JHASH) 7,3,3                                   CSP04550
1      IF (JSIGN*KSIGN) 7,8,8                                         CSP04560
7      ICOMP=1                                                        CSP04570
8      ICOMP=JSIGN*ICOMP                                              CSP04580
3      CALL NSIGN(JCARD,JLAST,JSIGN,KNOW)                             CSP04590
       CALL NSIGN(KCARD,KLAST,KSIGN,KNOW)                             CSP04600
       RETURN                                                         CSP04610
       END                                                            CSP04620



  1130 COMMERCIAL SUBROUTINE PACKAGE                                  PAGE 03

 VARIABLE ALLOCATIONS
  ICOMP=0000  JSIGN=0001  KSIGN=0002  KSTRT=0003  KNOW =0004  JHASH=0005

 STATEMENT ALLOCATIONS
  5    =0042  6    =0054  4    =005C  2    =0088  1    =00A2  7    =00A9  8    =00A0  3    =0084

 FEATURES SUPPORTED
  ONE WORD INTEGERS
  EXTENDED PRECISION

 CALLED SUBPROGRAMS
  NSIGN   SUBSC   SUBIN

 INTEGER CONSTANTS
      1=0008      0=0009

 CORE REQUIREMENTS FOR ICOMP
  COMMON       0  VARIABLES      8  PROGRAM     188

 END OF COMPILATION
```

```
// DUP                                                          CSP04630
*STORE      WS  UA  ICOMP                                       CSP04640
2ABB 000C

// FOR                                                          CSP04650
```



```
** 1130 COMMERCIAL SUBROUTINE PACKAGE                          CSP04660
* NAME NSIGN                                                   CSP04670
* ONE WORD INTEGERS                                            CSP04680
* EXTENDED PRECISION                                           CSP04690
* LIST ALL                                                     CSP04700
```



```
      SUBROUTINE NSIGN(JCARD,J,NEWS,NOLDS)                     CSP04710
      DIMENSION JCARD(80)                                      CSP04720
C-----SIGN OF DECIMAL INTEGER AT JCARD(J) IS SET +, - OR REVERSED AS  CSP04730
C-----NEWS IS +1, -1 OR 0.  NOLDS IS SET TO +1 OR -1 AS JCARD(J) WAS +  CSP04740
C-----OR -.                                                    CSP04750
      NOLDS=1                                                  CSP04760
      JTEST=JCARD(J)                                           CSP04770
      IF (JTEST) 1,2,2                                         CSP04780
1     NOLDS=-1                                                 CSP04790
2     IF (NEWS*JTEST) 3,4,5                                    CSP04800
4     IF (NEWS) 3,3,5                                          CSP04810
3     JTEST=-JTEST-1                                           CSP04820
5     JCARD(J)=JTEST                                           CSP04830
      RETURN                                                   CSP04840
      END                                                      CSP04850
```



```
VARIABLE ALLOCATIONS
 JTEST=0000

STATEMENT ALLOCATIONS
 1    =0023  2    =002B  4    =0031  3    =0035  5    =003C

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC   SUBIN

INTEGER CONSTANTS
     1=0002

CORE REQUIREMENTS FOR NSIGN
 COMMON       0  VARIABLES      2  PROGRAM      70

END OF COMPILATION

// DUP                                                          CSP04860
*STORE      WS  UA  NSIGN                                       CSP04870
2AC4 0005

// FOR                                                          CSP04880
```



```
** 1130 COMMERCIAL SUBROUTINE PACKAGE                          CSP04890
* NAME A1DEC                                                   CSP04900
* ONE WORD INTEGERS                                            CSP04910
* EXTENDED PRECISION                                           CSP04920
* LIST ALL                                                     CSP04930
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                                PAGE 02

      SUBROUTINE A1OEC(JCARO,J,JLAST,NER)                                          CSP04940
      DIMENSION JCARO(80)                                                          CSP04950
C-----CONVERT FIELD AT JCARO FROM A1 TO DECIMAL, CONVERTING BLANKS TO 0,          CSP04960
C-----IF NOT NUMERIC OR BLANK, SET ERROR INDICATOR TO SUBSCRIPT VALUE.            CSP04970
      CALL NZONE(JCARO,JLAST,4,JSIGN)                                              CSP04980
      DO 8 JNOW=J,JLAST                                                            CSP04990
      JTEST=JCARO(JNOW)                                                            CSP05000
      IF (JTEST) 2,3,3                                                             CSP05010
2     IF (JTEST+4032) 4,1,1                                                        CSP05020
4     NER=JNOW                                                                     CSP05030
      GO TO 8                                                                      CSP05040
3     IF (JTEST-16448) 4,5,4                                                       CSP05050
5     JTEST=-4032                                                                  CSP05060
1     JCARO(JNOW)=(JTEST+4032)/256                                                 CSP05070
8     CONTINUE                                                                     CSP05080
      IF (JSIGN-2) 6,7,6                                                           CSP05090
7     JCARO(JLAST)=-JCARO(JLAST)-1                                                 CSP05100
6     RETURN                                                                       CSP05110
      END                                                                          CSP05120


 1130 COMMERCIAL SUBROUTINE PACKAGE                                                PAGE 03

VARIABLE ALLOCATIONS
 JSIGN=0000   JNOW =0001   JTEST=0002

STATEMENT ALLOCATIONS
 2    =0032  4    =003B  3    =003E  5    =0044  1    =0049  8    =0057  7    =0065  6    =0071

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NZONE   SUBSC   SUBIN

INTEGER CONSTANTS
      4=0004   4032=0005  16448=0006    256=0007      2=0008      1=0009

CORE REQUIREMENTS FOR A1DEC
 COMMON       0  VARIABLES      4  PROGRAM    112

END OF COMPILATION



// DUP                                                                           CSP05130

*STORE      WS  UA  A1OEC                                                        CSP05140

2AC9 0008



// FOR                                                                           CSP05150



                                                                                 PAGE 01

** 1130 COMMERCIAL SUBROUTINE PACKAGE                                            CSP05160
* NAME DECA1                                                                     CSP05170
* ONE WORD INTEGERS                                                              CSP05180
* EXTENDED PRECISION                                                             CSP05190
* LIST ALL                                                                       CSP05200


  1130 COMMERCIAL SUBROUTINE PACKAGE                                             PAGE 02

      SUBROUTINE DECA1(JCARO,J,JLAST,NER)                                        CSP05210
      DIMENSION JCARO(80)                                                        CSP05220
C-----CONVERT FIELD AT JCARO FROM DECIMAL TO A1,                                 CSP05230
C-----IF NOT NUMERIC, SET ERROR INDICATOR TO SUBSCRIPT VALUE.                    CSP05240
      JSIGN=4                                                                    CSP05250
      IF (JCARO(JLAST)) 1,2,2                                                    CSP05260
1     JSIGN=2                                                                    CSP05270
      JCARO(JLAST)=-JCARO(JLAST)-1                                               CSP05280
2     DO 3 JNOW=J,JLAST                                                          CSP05290
      JTEST=JCARO(JNOW)                                                          CSP05300
      IF (JTEST) 4,5,5                                                           CSP05310
5     IF (JTEST-10) 6,4,4                                                        CSP05320
4     NER=JNOW                                                                   CSP05330
      GO TO 3                                                                    CSP05340
6     JCARO(JNOW)=256*JTEST-4032                                                 CSP05350
3     CONTINUE                                                                   CSP05360
      CALL NZONE(JCARO,JLAST,JSIGN,JNOW)                                         CSP05370
      RETURN                                                                     CSP05380
      END                                                                        CSP05390
```

```
 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 JSIGN=0000  JNOW =0001  JTEST=0002

STATEMENT ALLOCATIONS
 1    =002A  2    =003A  5    =0048  4    =0051  6    =0057       =0065

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 NZONE   SUBSC   SUBIN

INTEGER CONSTANTS
     4=0004      2=0005      1=0006     10=0007    256=0008   4032=0009

CORE REQUIREMENTS FOR DECA1
 COMMON       0  VARIABLES      4  PROGRAM    114

END OF COMPILATION


// DUP                                                                CSP05400
*STORE      WS  UA  DECA1                                              CSP05410
2A01 0008


// FOR                                                                CSP05420


                                                                       PAGE 01
** 1130 COMMERCIAL SUBROUTINE PACKAGE                                 CSP05430
* NAME CARRY                                                          CSP05440
* ONE WORD INTEGERS                                                   CSP05450
* EXTENDED PRECISION                                                  CSP05460
* LIST ALL                                                            CSP05470


 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 02

      SUBROUTINE CARRY(JCARD,J,JLAST,KARRY)                            CSP05480
      DIMENSION JCARD(80)                                              CSP05490
C-----RESOLVE ALL CARRIES IN FIELD AT JCARD, KARRY IS HIGH ORDER CARRY. CSP05500
      NCARY=0                                                          CSP05510
      JNOW=JLAST                                                       CSP05520
4     JTEST=JCARD(JNOW)+NCARY                                          CSP05530
      NCARY=JTEST/10                                                   CSP05540
      JTEST=JTEST-10*NCARY                                             CSP05550
      IF (JTEST) 1,2,2                                                 CSP05560
1     JTEST=JTEST+10                                                   CSP05570
      NCARY=NCARY-1                                                    CSP05580
2     JCARD(JNOW)=JTEST                                                CSP05590
      JNOW=JNOW-1                                                      CSP05600
      IF (JNOW-J) 3,4,4                                                CSP05610
3     KARRY=NCARY                                                      CSP05620
      RETURN                                                           CSP05630
      END                                                              CSP05640


 1130 COMMERCIAL SUBROUTINE PACKAGE                                    PAGE 03

VARIABLE ALLOCATIONS
 NCARY=0000  JNOW =0001  JTEST=0002

STATEMENT ALLOCATIONS
 4    =0018  1    =003C  2    =0048  3    =0050

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION

CALLED SUBPROGRAMS
 SUBSC   SUBIN

INTEGER CONSTANTS
     0=0004     10=0005      1=0006

CORE REQUIREMENTS FOR CARRY
 COMMON       0  VARIABLES      4  PROGRAM     96

END OF COMPILATION


// DUP                                                                CSP05650
*STORE      WS  UA  CARRY                                              CSP05660
2A09 0007


// ASM                                                                CSP05670
** IONO SUBROUTINE FOR 1130 COMMERCIAL SUBROUTINE PACKAGE          (ID) CSP05680
* NAME IONO                                                       (ID) CSP05690
* LIST (1132 PRINTER)                                                 CSP05700
```

```
                                                                      PAGE   1

0000     09595100           ENT     IONO       SUBROUTINE NAME              CSP05710
                   *CALL IOND    NO PARAMETERS                              CSP05720
                   *CALL IOND    ALLOWS I/O OPERATIONS TO END BEFORE A       CSP05730
                   *             PAUSE OR STOP IS ENTERED                   CSP05740
0000     0001      IOND  BSS     1          ARGUMENT ADDRESS                CSP05750
0001 00 74000032   IOPND MDX  L  50,0       ANY INTERRUPTS PENDING          CSP05760
0003 0  70FD             MDX     IOPNO                                      CSP05770
0004 01 4C800000   BACK  BSC  I  IONO                                       CSP05780
0006                     END                                               CSP05790

    NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                     CSP05800

*STORE      WS  UA  IONO                                                   CSP05810

2AE0 0002


// ASM                                                                     CSP05820
** PACK/UNPAC SUBROUTINES FOR 1130 COMMERCIAL SUBROUTINE PACKAGE      (IO) CSP05830
* NAME UNPAC                                                         (IO) CSP05840
* LIST (1132 PRINTER)                                                     CSP05850


                                                                      PAGE   1

0000     24557043           ENT     UNPAC UNPACK SUBROUTINE ENTRY POINT    CSP05860
                   *             CALL UNPAC(JCARD,J,JLAST,KCARD,K)          CSP05870
                   *             THE WORDS JCARD J  THROUGH                 CSP05880
                   *             JCARD JLAST  IN A2 FORMAT ARE              CSP05890
                   *             UNPACKED INTO KCARD K  IN A1 FORMAT.       CSP05900
0006     17043480           ENT     PACK PACK SUBROUTINE ENTRY POINT       CSP05910
                   *             CALL PACK(JCARD,J,JLAST,KCARD,K)           CSP05920
                   *             THE WORDS JCARD J  THROUGH                 CSP05930
                   *             JCARD JLAST  IN A1 FORMAT ARE PACKED       CSP05940
                   *             INTO KCARD K  IN A2 FORMAT.                CSP05950
0000 0  0000       UNPAC OC      0 ARGUMENT ADDRESS COMES IN HERE           CSP05960
0001 0  C003              LD      SW2 LOAD NOP INSTRUCTION                   CSP05970
0002 0  001E              STO     SWTCH STORE NOP AT SWITCH                  CSP05980
0003 0  7007              MDX     START COMPUTING                            CSP05990
0004 0  7008       SW1    MDX  X  ELSE-SWTCH-1 BRANCH TO ELSE                CSP06000
0005 0  7000       SW2    MDX  X  0 NOP INSTRUCTION                          CSP06010
0006 0  0000       PACK   DC      0 ARGUMENT ADDRESS COMES IN HERE           CSP06020
0007 0  C0FE              LD      PACK PICK UP ARGUMENT ADDRESS              CSP06030
0008 0  00F7              STO     UNPAC AND STORE IT IN UNPAC                CSP06040
0009 0  C0FA              LD      SW1 LOAD BRANCH TO ELSE                    CSP06050
000A 0  D016              STO     SWTCH STORE BRANCH AT SWITCH               CSP06060
000B 0  692F       START  STX   1  SAVE1&1 SAVE IR1                          CSP06070
000C 01 65800000          LDX  11 UNPAC PUT ARGUMENT ADDRESS IN IR1          CSP06080
000E 0  C100              LD    1 0 GET JCARD ADDRESS                      CSP06090
000F 0  802F              A       ONE ADD CONSTANT OF 1                    CSP06100
0010 00 95800001          S    11 1 SUBTRACT J VALUE                       CSP06110
0012 0  D000              STO     JCARD+1 CREATE JCARD(J) ADDRESS          CSP06120
0013 0  C103              LD    1 3 GET KCARD ADDRESS                      CSP06130
0014 0  802A              A       ONE ADD CONSTANT OF 1                    CSP06140
0015 00 95800004          S    11 4 SUBTRACT K VALUE                       CSP06150
0017 0  0006              STO     KCARD+1 CREATE KCARD(K) ADDRESS          CSP06160
0018 0  C100              LD    1 0 GET JCARD ADDRESS                      CSP06170
0019 0  8025              A       ONE ADD CONSTANT OF 1                    CSP06180
001A 00 95800002          S    11 2 SUBTRACT JLAST VALUE                   CSP06190
001C 0  0023              STO     JLAST CREATE JCARD JLAST  ADDRESS        CSP06200
001D 00 65000000   KCARD LDX  L1 0 PUT KCARD ADDRESS IN IR1                 CSP06210
001F 00 C4000000   JCARD LD   L  0 PICK UP JCARD(J)                         CSP06220
0021 0  7000       SWTCH MDX  X  0 SWITCH BETWEEN PACK AND UNPACK           CSP06230
0022 0  1888              SRT     8 SHIFT LOW ORDER BITS TO EXT            CSP06240
0023 0  1008              SLA     8 REPOSITION HIGH ORDER BITS             CSP06250
0024 0  EB19              OR      BMASK PUT BLANK IN LOW ORDER BITS        CSP06260
0025 0  0100              STO   1 0 PUT IN KCARD K                         CSP06270
0026 0  71FF              MDX   1 -1 DECREMENT KCARD ADDRESS               CSP06280
0027 0  1090              SLT     16 REPOSITION BITS FROM EXT              CSP06290
0028 0  E815              OR      BMASK PUT BLANK IN LOW ORDER BITS        CSP06300
0029 0  70D6              MDX     FINIS BRANCH AROUND PACK ROUTINE         CSP06310
002A 0  1B98       ELSE   SRT     24 SHIFT HIGH ORDER BITS INTO EXT        CSP06320
002B 01 74FF0020          MDX  L  JCARD+1,-1 DECREMENT JCARD ADDRESS       CSP06330
002D 01 C4800020          LD    1 JCARD+1 PICK UP JCARD(J+1)               CSP06340
002F 0  18C8              RTE     8 SHIFT IN BITS FROM EXT                 CSP06350
0030 0  D100       FINIS  STO   1 0 PUT IN KCARD K                         CSP06360
0031 01 74FF0020          MDX  L  JCARD+1,-1 DECREMENT JCARD ADDRESS       CSP06370
0033 0  71FF              MDX   1 -1 DECREMENT KCARD ADDRESS               CSP06380
0034 0  C0EB              LD      JCARD+1 GET JCARD(J) ADDRESS             CSP06390


                                                                      PAGE   2

0035 0  900A              S       JLAST SUBTRACT JCARD JLAST  ADDRESS      CSP06400
0036 01 4C10001F          BSC  L  JCARO,- CONTINUE IF DIFFERENCE & OR      CSP06410
0038 01 74050000          MDX  L  UNPAC,5 CREATE RETURN ADDRESS            CSP06420
003A 00 65000000   SAVE1  LDX  L1 0 RESTORE IR1                           CSP06430
003C 01 4C800000          BSC  I  UNPAC RETURN TO CALLING PROGRAM          CSP06440
003E 0  0040       BMASK  DC      /40 MASK 0000000001000000                CSP06450
003F 0  0001       ONE    DC      1 CONSTANT OF 1                          CSP06460
0040 0  0000       JLAST  DC      0 STORAGE FOR JCARD JLAST  ADDRESS       CSP06470
0042                      END                                              CSP06480

    NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                     CSP06490

*STORE      WS  UA  UNPAC                                                  CSP06500

2AE2 0005
```

```
// ASM                                                                  CSP06510
** READ AND PUNCH SUBROUTINES FOR 1130 CSP                        (ID) CSP06520
* NAME READ                                                       (ID) CSP06530
* LIST (1132 PRINTER)                                                 CSP06540


                                                                      PAGE  1
0053    19141100          ENT     READ      SUBROUTINE ENTRY POINT    CSP06550
                        * CALL READ (JCARD, J, JLAST, NERR1)          CSP06560
                        * READ COLUMNS FROM BEGINNING OF CARD INTO JCARD(J) CSP06570
                        * THROUGH JCARD(JLAST). PUT ERROR PARAMETER IN CSP06580
                        * NERR1.                                      CSP06590
008C    179150C8          ENT     PUNCH     SUBROUTINE ENTRY POINT    CSP06600
                        * CALL PUNCH (JCARD, J, JLAST, NERR2)         CSP06610
                        * PUNCH JCARD(J) THROUGH JCARD(JLAST) INTO THE CSP06620
                        * BEGINNING OF A CARD. PUT ERROR PARAMETER INTO CSP06630
                        * NERR2.                                      CSP06640
0000 0  0000      JCARD DC      0         JCARD J  ADDRESS          CSP06650
0001    0051      AREA  BSS     81        I/O AREA BUFFER           CSP06660
0052 0  0000      FLAG  DC      0         ERROR INDICATOR           CSP06670
0053 0  0000      READ  DC      0         FIRST ARGUMENT ADDRESS    CSP06680
0054 0  691B            STX   1 SAVE161   SAVE IR1                  CSP06690
0055 01 65800053        LDX   I1 READ     GET 1ST ARGUMENT ADDRESS  CSP06700
0057 0  4022            BSI     SETUP     GO TO SETUP               CSP06710
0058 20 03059131        LIBF    CARD1     CALL CARD READ ROUTINE    CSP06720
0059 0  1000            DC      /1000     READ                      CSP06730
005A I  0001            DC      AREA      AREA PARAMETER            CSP06740
0058 1  0073            DC      ERROR     ERROR PARAMETER           CSP06750
005C 20 225C5144  CONVT LIBF    SPEED     CALL CONVERSION ROUTINE   CSP06760
0050 0  0010            DC      /0010     CARD CODE TO EBCDIC       CSP06770
005E I  0002            DC      AREA61    FROM AREA                 CSP06780
005F 0  0000      JLAS1 DC      0         TO JCARD JLAST            CSP06790
0060 0  0000      CNT1  DC      0         CHARACTER COUNT           CSP06800
0061 0  C0F0            LD      FLAG      ERROR INDICATOR           CSP06810
0062 01 4C180067        BSC   L FINAL,6-  ALL DONE IF ZERO          CSP06820
0064 0  1810            SRA     16        CLEAR ACC                 CSP06830
0065 0  00EC            STO     FLAG      CLEAR THE INDICATOR       CSP06840
0066 0  70F5            MDX     CONVT     CONVERT AGAIN             CSP06850
0067 20 22989547  FINAL LIBF    SWING     REVERSE THE ARRAY         CSP06860
0068 1  0000            DC      JCARD     FROM JCARD J              CSP06870
0069 1  005F            DC      JLAS1     TO JCARD JLAST            CSP06880
006A 20 03059131  TEST  LIBF    CARD1     CALL BUSY TEST ROUTINE    CSP06890
0068 0  0000            DC      /0000     BUSY TEST PARAMETER       CSP06900
006C 0  70FD            MDX     TEST      REPEAT IF BUSY            CSP06910
006D 0  7104            MDX     I 4       INCREMENT 4 ARGUMENTS     CSP06920
006E 0  6903            STX     1 DONE61  STORE IR1                 CSP06930
006F 00 65000000  SAVE1 LDX   L1 0         RESTORE IR1               CSP06940
0071 00 4C000000  DONE  BSC   L  0         RETURN TO CALLING PROGRAM CSP06950
0073 0  0000      ERROR DC      0         START OF ERROR ROUTINE    CSP06960
0074 00 D4000000  ERR   STO   L  0         STORE ACC IN ERROR WORD   CSP06970
0076 01 74010052        MDX   L  FLAG,1   SET THE FLAG INDICATOR     CSP06980
0078 01 4C800073        BSC   I  ERROR     RETURN TO INTERRUPT PROGRM CSP06990
007A 0  0000      SETUP DC      0         START OF SETUP ROUTINE    CSP07000
0078 20 01667880        LIBF    ARGS      CALL ARGS SUBPROGRAM      CSP07010
007C 1  0000            DC      JCARD     GET JCARD J  ADDRESS      CSP07020
007D 1  005F            DC      JLAS1     GET JCARD JLAST  ADDRESS  CSP07030
007E 1  0001            DC      AREA      GET CHARACTER COUNT       CSP07040
007F 0  0050            DC      80        MAX CHARACTER COUNT       CSP07050
0080 0  C00E            LD      JLAS1     DISTRIBUTE JCARD JLAST    CSP07060
0081 0  0014            STO     JLAS2     INTO JLAS2                CSP07070


                                                                      PAGE  2
0082 01 C4000001        LD    L  AREA      DISTRIBUTE COUNT          CSP07080
0084 0  00D8            STO     CNT1      INTO CNT1                 CSP07090
0085 0  D012            STO     CNT2      AND CNT2                  CSP07100
0086 0  C103            LD    1 3         GET ERROR WORD ADDRESS    CSP07110
0087 0  D0ED            STO     ERR61     STORE INSIDE ERROR ROUTINE CSP07120
0088 0  1810            SRA     16        CLEAR ACC                 CSP07130
0089 0  00C8            STO     FLAG      CLEAR ERROR INDICATOR     CSP07140
008A 01 4C80007A        BSC   I  SETUP     RETURN TO CALLING PROG    CSP07150
008C 0  0000      PUNCH DC      0         PUNCH ROUTINE STARTS HERE CSP07160
008D 0  69E2            STX   1 SAVE161   SAVE IR1                  CSP07170
008E 01 6580008C        LDX   I1 PUNCH    LOAD 1ST ARGUMENT ADDRESS CSP07180
0090 0  40E9            BSI     SETUP     GO TO SETUP ROUTINE       CSP07190
0091 20 22989547        LIBF    SWING     CALL REVERSE ARRAY        CSP07200
0092 1  0000            DC      JCARD     FROM JCARD J              CSP07210
0093 1  005F            DC      JLAS1     TO JCARD JLAST            CSP07220
0094 20 225C5144        LIBF    SPEED     CALL CONVERSION ROUTINE   CSP07230
0095 0  0011            DC      /0011     FROM EBCDIC TO CARD CODE  CSP07240
0096 0  0000      JLAS2 DC      0         FROM JCARD JLAST          CSP07250
0097 1  0002            DC      AREA61    TO THE I/O AREA BUFFER    CSP07260
0098 0  0000      CNT2  DC      0         CHARACTER COUNT           CSP07270
0099 20 03059131        LIBF    CARD1     CALL PUNCH ROUTINE        CSP07280
009A 0  2000            DC      /2000     PUNCH                     CSP07290
0098 I  0001            DC      AREA      I/O AREA BUFFER           CSP07300
009C 1  0073            DC      ERROR     ERROR PARAMETER           CSP07310
009D 0  70C9            MDX     FINAL     ALL THROUGH, GO TO FINAL  CSP07320
009E                    END               END OF READ SUBPROGRAM    CSP07330

   NO ERRORS IN ABOVE ASSEMBLY.


// DUP                                                                  CSP07340

*STORE      WS  UA  READ                                                CSP07350

2AE7 0006


// ASM                                                                  CSP07360
** TYPE AND KEYBD SUBROUTINES FOR 1130 CSP                        (ID) CSP07370
* NAME TYPER                                                      (ID) CSP07380
* LIST (1132 PRINTER)                                                 CSP07390
```

```
003F    23A17159          ENT     TYPER       SUBROUTINE ENTRY POINT      CSP07400
                  * CALL TYPE (JCARD, J, JLAST)                           CSP07410
                  * TYPE JCARO(J) THROUGH JCARO(JLAST)                    CSP07420
0069    12168084          ENT     KEYBD       SUBROUTINE ENTRY POINT      CSP07430
                  * CALL KEYBD (JCARD, J, JLAST)                          CSP07440
                  * ENTER AT KEYBOARD JCARD(J) THROUGH JCARD(JLAST)       CSP07450
0000 0  0001      ONE   DC      1           CONSTANT OF 1                 CSP07460
0001 0  0000      JCARO DC      0           JCARO J  AODRESS              CSP07470
0002    0030      AREA  BSS     61          I/O AREA BUFFER               CSP07480
003F 0  0000      TYPER DC      0           FIRST ARGUMENT AODR HERE      CSP07490
0040 0  691A            STX   1 SAVE161     SAVE IR1                      CSP07500
0041 0  6178            LDX   1 120         PUT 120 IN IR1                CSP07510
0042 0  6923            STX   1 MAXCH       STORE IT AS MAX CHARS         CSP07520
0043 01 6580003F        LDX   I1 TYPER      PUT FIRST ADDR IN IR1         CSP07530
0045 0  4018            BSI     SETUP       GO TO SETUP                   CSP07540
0046 0  C0B8            LD      AREA        GET CHARACTER COUNT           CSP07550
0047 0  80B8            A       ONE         HALF AOJUST IT AND            CSP07560
004B 0  1801            SRA     1           DIVIOE IT BY TWO              CSP07570
0049 0  D0B8            STO     AREA        AND REPLACE IT                CSP07580
004A 0  1001            SLA     1           OOUBLE IT                     CSP07590
004B 0  D008            STO     CNT1        ANO PUT IT IN CNT1            CSP07600
004C 20 195C1002        LIBF    RPACK       CALL REVERSE PACK ROUTINE     CSP07610
004D 1  0001            DC      JCARO       FROM JCARD J                  CSP07620
004E 1  0083            OC      JLAST       TO JCARD JLAST                CSP07630
004F 1  0003            DC      AREA61      PACK INTO I/O AREA            CSP07640
0050 20 05097663        LIBF    EBPRT       CALL CONVERSION ROUTINE       CSP07650
0051 0  0000            DC      /0000       FROM EBCDIC                   CSP07660
0052 1  0003            DC      AREA61      TO PRINTER COOE,              CSP07670
0053 1  0003            DC      AREA61      ALL IN THE I/O AREA           CSP07680
0054 0  0000      CNT1  DC      0           HALF AOJSTD CHARACTER CNT     CSP07690
0055 20 23A17170        LIBF    TYPEO       CALL TYPE ROUTINE             CSP07700
0056 0  2000            DC      /2000       TYPE PARAMETER                CSP07710
0057 1  0002            DC      AREA        I/O AREA BUFFER               CSP07720
005B 0  7103      FINAL MOX     1 3         INCREMENT OVER 3 ARGUMENTS    CSP07730
0059 0  6903            STX   1 DONE61      STORE IR1                     CSP07740
005A 00 65000000  SAVE1 LDX   L1 0         RESTORE IR1                   CSP07750
005C 00 4C000000  OONE  BSC   L  0         RETURN TO CALLING PROGRAM     CSP07760
005E 0  0000      SFTUP DC      0           START OF SETUP ROUTINE        CSP07770
005F 20 23A17170  TEST  LIBF    TYPEO       CALL BUSY TEST ROUTINE        CSP07780
0060 0  0000            DC      /0000       BUSY TEST PARAMETER           CSP07790
0061 0  70FD            MDX     TEST        REPEAT TEST IF BUSY           CSP07800
0062 20 01647880        LIBF    ARGS        CALL ARGS ROUTINE             CSP07810
0063 1  0001            DC      JCARO       1ST ARGUMENT TO JCARD J       CSP07820
0064 1  0083            DC      JLAST       TO JCARD JLAST                CSP07830
0065 1  0002            DC      AREA        TO CHARACTER COUNT            CSP07840
0066 0  0000      MAXCH DC      0           MAXIMUM NUMBER OF CHARS       CSP07850
0067 01 4C80005E        BSC   1 SETUP       END OF SETUP, RETURN          CSP07860
0069 0  0000      KEYBD DC      0           START OF KEYBOARO ROUTINE     CSP07870
006A 0  69F0            STX   1 SAVE161     SAVE IR1                      CSP07880
006B 0  613C            LDX   1 60          PUT BUFFER LENGTH IN IR1      CSP07890
006C 0  69F9            STX   1 MAXCH       60 IS MAX NO OF CHARS         CSP07900
006D 01 65800069        LOX   I1 KEYBO      1ST ARGUMENT AOOR IN IR1      CSP07910
006F 0  40EE            BSI     SETUP       GO TO SETUP                   CSP07920
```



```
0070 0  613C            LDX   1 60          PUT BUFFER LENGTH IN IR1      CSP07930
0071 0  1810            SRA     16          CLEAR THE ACC                 CSP07940
0072 01 D5000002  CLEAR STO   L1 AREA       CLEAR THE I/O BUFFER          CSP07950
0074 0  71FF            MDX   1 -1          DECREMENT IR1                 CSP07960
0075 0  70FC            MDX     CLEAR       AND CONTINUE CLEARING         CSP07970
0076 01 65800069        LDX   I1 KEYBO      1ST ARGUMENT ADOR IN IR1      CSP07980
007B 0  C089            LD      AREA        PUT CHARACTER COUNT           CSP07990
0079 0  000A            STO     CNT2        IN CNT2                       CSP08000
007A 20 23A17170        LIBF    TYPEO       CALL KEYBOARO ROUTINE         CSP08010
007B 0  1000            DC      /1000       KEYBOARO PARAMETER            CSP08020
007C 1  0002            DC      AREA        I/O AREA BUFFER               CSP08030
007D 20 23A17170  TEST1 LIBF    TYPEO       CALL BUSY TEST ROUTINE        CSP08040
007E 0  0000            DC      /0000       BUSY TEST PARAMETER           CSP08050
007F 0  70FD            MDX     TEST1       REPEAT TEST IF BUSY           CSP08060
0080 20 225C5144        LIBF    SPEED       CALL CONVERSION ROUTINE       CSP08070
0081 0  0010            DC      /0010       CARO CODE TO EBCOIC           CSP08080
0082 1  0003            DC      AREA61      FROM THE I/O AREA BUFFER      CSP08090
0083 0  0000      JLAST DC      0           TO JCARD JLAST                CSP08100
0084 0  0000      CNT2  DC      0           CHARACTER COUNT               CSP08110
0085 20 22989547        LIBF    SWING       CALL REVERSE ARRAY            CSP08120
0086 1  0001            DC      JCARO       REVERSE FROM JCARD J          CSP08130
0087 1  0083            OC      JLAST       TO JCARD JLAST                CSP08140
0088 0  70CF            MOX     FINAL       ALL THROUGH, GO TO FINAL      CSP08150
008A                    ENO                 END OF TYPE SUBPROGRAM        CSP08160
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP08170
*STORE      WS  UA  TYPER                                               CSP08180
2AED 0006
```

```
// ASM                                                                  CSP08190
** PRINT AND SKIP SUBROUTINES FOR 1130 CSP                    (ID) CSP08200
* NAME PRINT                                                  (IO) CSP08210
* LIST (1132 PRINTER)                                             CSP08220
```

```
0041    17649563          ENT     PRINT      SUBROUTINE ENTRY POINT      CSP08230
                       * CALL PRINT (JCARO, J, JLAST, NERR3)             CSP08240
                       * PRINT JCARD(J) THROUGH JCARD(JLAST) ON THE      CSP08250
                       * 1132 PRINTER. PUT ERROR PARAMETER IN NERR3.     CSP08260
0069    224895C0          ENT     SKIP       SUBROUTINE ENTRY POINT      CSP08270
                       * CALL SKIP(N)                                    CSP08280
                       * EXECUTE CONTROL FUNCTION SPECIFIED BY INTEGER N CSP08290
0000 0  0001       ONE    DC      1          CONSTANT OF 1               CSP08300
0001 0  2000       SPACE  DC      /2000      PRINT FUNCTION WITH SPACE   CSP08310
0002 0  0000       JCARD  DC      0          JCARD J  ADDRESS            CSP08320
0003 0  0000       JLAST  DC      0          JCARD JLAST  ADDRESS        CSP08330
0004    003D       AREA   BSS     61         WORD COUNT & PRINT AREA     CSP08340
0041 0  0000       PRINT  DC      0          ADDRESS OF 1ST ARGUMENT     CSP08350
0042 20 176558F1   TEST   LIBF    PRNT1      CALL BUSY TEST ROUTINE      CSP08360
0043 0  0000              DC      /0000      BUSY TEST PARAMETER         CSP08370
0044 0  70FD              MDX     TEST       REPEAT TEST IF BUSY         CSP08380
0045 0  691A              STX   1 SAVE161    STORE IR1                   CSP08390
0046 01 65800041          LDX   I1 PRINT     LOAD 1ST ARGUMENT ADDRESS   CSP08400
0048 20 01647880          LIBF    ARGS       CALL ARGS ROUTINE           CSP08410
0049 1  0002              DC      JCARD      JCARD J  PICKED UP          CSP08420
004A 1  0003              DC      JLAST      JCARD JLAST  PICKED UP     CSP08430
004B 1  0004              DC      AREA       CHARACTER COUNT PICKED UP   CSP08440
004C 0  0078              DC      120        MAX CHARACTER COUNT         CSP08450
004D 0  C086              LD      AREA       GET CHARACTER COUNT         CSP08460
004E 0  80B1              A       ONE        HALF ADJUST                 CSP08470
004F 0  1801              SRA     1          DIVIDE BY TWO               CSP08480
0050 0  D0B3              STO     AREA       STORE WORD COUNT            CSP08490
0051 0  C103              LD    1 3          GET ERROR WORD ADDRESS      CSP08500
0052 0  D012              STO     ERR61      STORE IT IN ERROR ROUTINE   CSP08510
0053 20 195C10D2          LIBF    RPACK      CALL REVERSE PACK ROUTINE   CSP08520
0054 1  0002              DC      JCARD      JCARD J  ADDRESS            CSP08530
0055 1  0003              DC      JLAST      JCARD JLAST  ADDRESS        CSP08540
0056 1  0005              DC      AREA61     PACK INTO I/O AREA          CSP08550
0057 20 176558F1          LIBF    PRNT1      CALL PRINT ROUTINE          CSP08560
0058 0  2000       WRITE  DC      /2000      PRINT PARAMETER             CSP08570
0059 1  0004              DC      AREA       I/O AREA BUFFER             CSP08580
005A 1  0063              DC      ERROR      ERROR PARAMETER             CSP08590
005B 0  C0A5              LD      SPACE      LOAD PRINT WITH SPACE       CSP08600
005C 0  00FB              STO     WRITE      STORE IN PRINT PARAMETER    CSP08610
005D 0  7104              MDX   1 4          INCREMENT OVER 4 ARGUMENTS  CSP08620
005E 0  6903              STX   1 DONE161    STORE IR1                   CSP08630
005F 00 65000000   SAVE1  LDX   L1 0         RELOAD OR RESTORE IR1       CSP08640
0061 00 4C000000   DONE1  BSC   L  0         RETURN TO CALLING PROGRAM   CSP08650
0063 0  0000       ERROR  DC      0          RETURN ADDRESS GOES HERE    CSP08660
0064 00 D4000000   ERR    STO   L  0         STORE ACC IN ERROR PARAM    CSP08670
0066 0  1B10              SRA     16         CLEAR ACC                   CSP08680
0067 01 4C800063          BSC   I  ERROR     RETURN TO PRNT1 PROGRAM     CSP08690
0069 0  0000       SKIP   DC      0          ADDRESS OF ARGUMENT ADDR    CSP08700
006A 01 C4800069          LD    I  SK1F      GET ARGUMENT ADDRESS        CSP08710
006C 0  D001              STO     ARG61      DROP IT AND                 CSP08720
006D 00 C4000000   ARG    LD    L  0         GET ARGUMENT                CSP08730
006F 01 4C300074          BSC   L  NOSUP+-2  GO TO NOSUPPRESSION IF 6    CSP08740
0071 0  C009              LD      NOSPC      SET UP SPACE SUPPRESSION    CSP08750
0072 0  D0E5              STO     WRITE      CHANGE PRINT FUNCTION       CSP08760
```



```
0073 0  7003              MDX     DONE       GO TO RETURN                CSP08770
0074 0  D001       NOSUP  STO     CNTRL      SET UP COMMAND              CSP08780
0075 20 176558F1          LIBF    PRNTI      CALL THE PRNT ROUTINE       CSP08790
0076 0  3000       CNTRL  DC      /3000      CARRIAGE COMMAND WORD       CSP08800
0077 01 74010069   DONE   MDX   L  SK1P+1    ADJUST RETURN ADDRESS       CSP08810
0079 01 4C800069          BSC   I  SKIP      RETURN TO CALLING PROGRAM   CSP08820
007B 0  2010       NOSPC  DC      /2010      SUPPRESS SPACE COMMAND      CSP08830
007C                      END                END OF PRINT SUBPROGRAM     CSP08840
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// DUP                                                                  CSP08850
*STORE      WS  UA  PRINT                                               CSP08860
2AF3 0005
```

```
// ASM                                                                  CSP08870
** ARGS, RPACK AND SWING SUBROUTINES FOR 1130 CSP                  (IO) CSP08880
* LIST (1132 PRINTER)                                                  CSP08890
* NAME ARGS                                                       (ID) CSP08900
```

```
                         LI8R                    LI8F TYPE ROUTINES FOLLOW   CSP08910
                    * THESE SUBROUTINES CANNOT 8E CALLED FROM FORTRAN       CSP08920
0002    01647880            ENT     ARGS        SUBROUTINE ENTRY POINT      CSP08930
                    * ARGS GETS THE ARGUMENT FOR THE I/O ROUTINES           CSP08940
0030    195C1002            ENT     RPACK       SUBROUTINE ENTRY POINT      CSP08950
                    * RPACK REVERSES AND PACKS E8CDIC STRINGS               CSP08960
0D4F    22989547            ENT     SWING       SUBRDUTINE ENTRY POINT      CSP08970
                    * SWING REVERSES AN E8CDIC STRING                       CSP08980
0000 0  0001        ONE     DC      1           CONSTANT OF I               CSP08990
0001 0  0D00        JLAST   DC      0           JCARD JLAST  ADDRESS         CSP09000
0002 0  6A2A        ARGS    STX   2 SAVE2&1     ARGS ROUTINE STARTS HERE    CSP09010
0003 00 66800000            LDK   I2 0          GET 1ST ARGUMENT ADDR       CSP09020
0D05 0  C100                LD    1 0          GET JCARD ADDR              CSP09030
0006 00 95800002            S     I1 2         SUBTRACT JLAST VALUE        CSP09040
0008 0  80F7                A       ONE        ADD ONE                     CSP09050
0009 00 D6800001            STO   I2 1         STORE IN 2ND ARG            CSP09060
0D08 0  C100                LD    1 0          GET JCARD ADDR              CSP09070
000C 00 95800001            S     I1 1         SUBTRACT J VALUE            CSP09080
0D0E 0  80F1                A       ONE        ADD ONE                     CSP09090
000F 00 D6800000            STO   I2 0         STORE IN 1ST ARG            CSP09100
0011 00 96800001            S     I2 1         SUBTRACT JLAST ADDR         CSP09110
0013 0  80EC                A       ONE        ADD ONE                     CSP09120
0D14 01 4C080018            BSC   L  ERDR1,+   CHECK FDR NEG OR 0 CHARS    CSP0913D
0D16 0  9203                S     2 3          DK. SUBTRACT MAK CHARS      CSP09140
0017 01 4C300021            BSC   L  ERROR,-Z  CHECK MORE THAN MAX CHARS    CSP09150
0019 0  8203                A     2 3          ADD MAK CHARS BACK          CSP09160
001A 0  7000                MDX     OK         ADDRESSES OK                CSP09170
0018 00 C6800000    ERORI   LD    I2 0         PICK UP JCARD(J)            CSP09180
001D 00 06800001            STO   I2 1         AND STORE IN JCARD(JLAST)   CSP09190
001F 0  C0E0                LO      DNE        SET UP CHAR COUNT OF 1      CSP09200
0020 0  7007                MDK     OK         GO TO STORE CHAR COUNT      CSP0921D
0021 00 C680D000    ERROR   LD    I2 0         PICK UP JCARD(J)            CSP09220
0023 0  9203                S     2 3          AND CALCULATE JCARD(JLAST)  CSP09230
0024 0  80D8                A       ONE        TO BE JCARD(J+MAX-1)        CSP09240
0025 00 06800001            STO   I2 1         STORE AODR IN JCARD(JLAST)  CSP09250
0027 0  C203                LD    2 3          LOAD CHARACTER COUNT        CSP09260
0028 00 06800002    OK      STO   I2 2         STORE CHARACTER COUNT       CSP09270
002A 0  7204                MDK   2 4          CREATE RETURN AODR          CSP09280
0D28 0  6A03        LAST    STX   2 DOME&1     STORE RETURN ADDRESS        CSP09290
0D2C 00 66000000    SAVE2   LDX   L2 0         RESTORE IR2                 CSP09300
002E 00 4C000000    DDNE    BSC   L  0          RETURN TO CALLING PROGRAM   CSP09310
0030 0  6AFC        RPACK   STK     2 SAVE2&1  RPACK ROUTINE STARTS HERE    CSP09320
0031 00 66800000            LDK   I2 0         GET 1ST ARGUMENT ADDRESS    CSP09330
0033 00 C6800000            LO    I2 0         GET JCARD ADDR              CSP09340
0035 0  D006                STO     JCARD&1    INITIALIZE JCARD ADDRESS    CSP09350
0036 00 C6800001            LD    I2 1         GET SECOMD ARGUMENT ADDR    CSP09360
0038 0  D0C8                STO     JLAST      INITIALIZE JCARD JLAST      CSP09370
0039 0  C202                LO    2 2          GET AREA ADDRESS            CSP09380
003A 0  D009                STO     KCARD&1    INITIALIZE PACK TO ADDRESS  CSP09390
003B 00 C4000000    JCARD   LD    L  0         LOAD FIRST CHARACTER        CSP09400
003D 0  1898                SRT     24         SHIFT INTO EXT              CSP09410
003E 01 74FF003C            MDK   L  JCARD&1,-1 DECREMENT ADDRESS           CSP0942D
0040 01 C480003C            LD    I  JCARD&1    GET SECOND CHARACTER        CSP09430
0042 0  18C8                RTE     8          SHIFT RIGHT, RETRIEVE EKT   CSP09440
```



```
0043 00 D4000000    KCARD   STO   L  0         STORE IN AREA               CSP09450
0045 01 74FF003C            MDK   L  JCARD&1,-1 DECREMENT ADDRESS           CSP09460
0047 01 74010044            MDX   L  KCARD&1,&1 INCREMENT AREA ADDRESS      CSP09470
0049 0  C0F2                LD      JCARD&1    GET ENDING ADDRESS          CSP09480
004A 0  90B6                S       JLAST      SUBTRACT JCARD JLAST  AODR CSP09490
004B 01 4C10003B            BSC   L  JCARD,=   REPEAT IF NDT MINUS         CSP09500
0D4D 0  7203                MDK   2 3          INCREMENT OVER 3 ARGS       CSP09510
004E 0  70DC                MDX     LAST       ALL THRDUGH, GO TO LAST     CSP09520
004F 0  6ADD        SWING   STK   2 SAVE2&1    SWIMG ARRAY END FDR END     CSP09530
0050 00 66800000            LDK   I2 0         GET 1ST ARGUMEMT ADDRESS    CSP09540
0052 00 C6800000            LD    I2 0         GET FIRST ARGUMENT          CSP09550
0054 0  D007                STO     8ACK&1     STORE AT BACK ADDRESS       CSP09560
0055 00 C6800001            LD    I2 1         GET 2ND ARGUMENT            CSP09570
0057 0  D001                STO     FROMT&1    STORE AT FRONT ADDRESS      CSP0958D
0058 00 C4000000    FRDNT   LD    L  0         GET WORD FROM FRONT         CSP09590
005A 0  1890                SRT     16         PUT IT IN THE EKT           CSP09600
D05B 00 C4000000    BACK    LD    L  0         GET A WORD FROM THE BACK    CSP09610
005D 0  E810                OR      HEK40      OR IN AN EBCDIC BLANK       CSP09620
005E 01 D4800059            STO   I  FRONT&1    PUT IT IN THE FRDNT         CSP09630
0060 0  1090                SLT     16         RETRIEVE THE EKT            CSP09640
D061 0  E80C                OR      HEX40      OR IN AN EBCDIC BLANK       CSP09650
0062 01 D480005C            STO   I  BACK&1     PUT IT IN THE BACK          CSP09660
0064 01 74010059            MDK   L  FRONT&1,&1 INCREMENT THE FRONT ADDR    CSP09670
0066 01 74FF005C            MDX   L  BACK&I,-1  DECREMENT THE BACK ADDR     CSP096B0
0068 0  C0F0                LD      FRONT&1    GET THE FRONT ADDRESS       CSP09690
0069 0  90F2                S       BACK+1     SUBTRACT THE 8ACK AODRESS   CSP097D0
006A 01 4C080058            BSC   L  FRONT,&   REPEAT IF MINUS             CSP09710
006C 0  7202                MDK   2 2          INCREMENT OVER 2 ARGS       CSP09720
006D 0  70BD                MDK     LAST       ALL THROUGH, GO TO LAST     CSP09730
006E 0  0040        MEX40   DC      /0040      E8CDIC BLANK COOE           CSP09740
0070                        END                END OF ARGS SUBPROGRAM      CSP09750
```

NO ERRORS IN A8OVE ASSEM8LY.

```
// DUP                                                                      CSP09760
*STORE      WS  UA  ARGS                                                    CSP09770
2AF8 0008
```

APPENDIX

CORE ALLOCATION

To calculate the core requirements, sum the number of words for all routines used. If NZONE, CARRY, NSIGN, SERVICE, WHOLE, ADD, and/or FILL are not included in the first sum, and they are CALLed by a routine in the first sum, add their number of words to the first sum. Then calculate the Reference core requirements. Keeping in mind that no matter how many times a Reference is used, it should be considered only once, sum the core requirements of all References used. Add this sum to the first sum. The resulting total is the core requirement for the 1130 Commercial Subroutine Package. Notice that the FORTRAN subroutines a, b, c, and d will also be used by most FORTRAN programs and so will be present whether the package is used or not.

Routine Name	Number of Words	CALLS	Reference
A1DEC	116	NZONE	a
ADD	202	NSIGN, CARRY, FILL	a
CARRY	100		a
DECA1	118	NZONE	a
DIV	354	NSIGN, CARRY, FILL	a
EDIT	302	NZONE, FILL	a
FILL	36		a
GET	148	NZONE	a,b,c
ICOMP	196	NSIGN	a
IOND	6		None
MOVE	56		a
MPY	240	NSIGN, CARRY, FILL	a
NCOMP	76		a
NSIGN	72		a
NZONE	136		a
PACK/UNPAC	66		None
PRINT/SKIP	124	SERVICE	e
PUT	152	NZONE, WHOLE	a,b,d
READ/PUNCH	158	SERVICE	f,h
STACK	6		None

Routine Name	Number of Words	CALLS	Reference
SUB	48	NSIGN, ADD	a
TYPER/KEYBD	136	SERVICE	g,h
WHOLE	34		None
SERVICE	112		None
TOTAL	2,994		

References

a) 62 (SUBSC, SUBIN)

b) 342 (EADD, EMPY, ESTO, FLOAT)

c) 8 (SNR)

d) 74 (EABS, ESBR, IFIX)

e) 404 (PRNT1)

f) 264 (CARD1)

g) 638 (TYPE0, EBPRT)

h) 360 (SPEED, ILS04)

EBCDIC CHARACTERS AND DECIMAL EQUIVALENTS

A	-16064	S	-7616	blank	16448
B	-15808	T	-7360	. (period)	19264
C	-15552	U	-7104	< (less than)	19520
D	-15296	V	-6848	(	19776
E	-15040	W	-6592	+	20032
F	-14784	X	-6336	&	20544
G	-14528	Y	-6080	$	23360
H	-14272	Z	-5824	*	23616
I	-14016	0	-4032	)	23872
J	-11968	1	-3776	- (minus)	24640
K	-11712	2	-3520	/	24896
L	-11456	3	-3264	,	27456
M	-11200	4	-3008	%	27712
N	-10944	5	-2752	#	31552
O	-10688	6	-2496	@	31808
P	-10432	7	-2240	' (apostrophe)	32064
Q	-10176	8	-1984	=	32320
R	-9920	9	-1728		

OPERATING INSTRUCTIONS

The procedures set forth in IBM 1130 Card/Paper Tape Programming System Operator's Guide (C26-3629) and in IBM 1130 DISK Monitor System Reference Manual (C26-3750) should be followed to execute the sample problems and all user-written programs.

In addition, to execute sample problems 1 and 3, the switch settings on the console are as follows:

Switch	Position and meaning
0	up = 1132 Printer, down = console printer
1-15	no meaning

There are no switch settings for sample problem 2, but the 1132 Printer is required.

HALT LISTING

Conditions A and B (see list below) have the following meaning:

A Device not ready.

B Internal subroutine error. Rerun job. If error persists, verify that the subroutine deck is accurate, using the listings in this manual. If the deck is the same, contact your local IBM representative. Save all output.

IAR	Accumulator (hex)	Device	Condition
41	1xx0	1442 Card Read Punch	A
41	1xx1	1442 Card Read Punch	B
41	2xx0	Console printer or keyboard	A
41	2xx1	Console printer or keyboard	B
41	6xx0	1132 Printer	A
41	6xx1	1132 Printer	B

BIBLIOGRAPHY

IBM 1130 Functional Characteristics (A26-5881)

Core Requirements for 1130 FORTRAN (C20-1641)

1130 FORTRAN Programming Techniques (C20-1642)

IBM 1130 Card/Paper Tape Programming Systems Operator's Guide (C26-3629)

IBM 1130 DISK Monitor System Reference Manual (C26-3750)

IBM 1130 Assembler Language (C26-5927)

IBM 1130 Subroutine Library (C26-5929)

IBM 1130 FORTRAN Language (C26-5933)

READER'S COMMENT FORM

1130 Commercial Subroutine Package (1130-SE-25X) H20-0241-2
Version 2, Program Reference Manual

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold fold

fold fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Technical Publications

fold fold

Printed in U.S.A. H20-0241-2

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Printed in U.S.A. H20-0241-2

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)